

中国计算机学会教育专业委员会 推荐
全国高等学校计算机教育研究会 出版
高等学校规划教材

人工智能基础

邵军力 张 景 魏长华 编著

计算机学科教学计划 1993



电子工业出版社

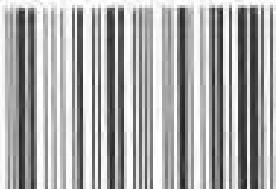
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL: <http://www.phei.com.cn>

- 1. 计算机导论（王玉龙编）
- 2. 数字逻辑与数字系统（王永军等编）
- 3. 电路与电子学（王文铸等编）
- 4. 离散数学（朱一清编）
- 5. 程序设计语言与编译（龚天富等编）
- 6. 计算机组装原理与汇编语言程序设计（徐远桢等编）
- 7. 算法与数据结构（傅清祥等编）
- 8. 数据通信与计算机网络（杨心强等编）
- 9. 人工智能基础（邵军力等编）
- 10. 计算机系统结构（张吉峰等编）
- 11. 计算机操作系统（刘乃琦等编）
- 12. 软件工程（杨文龙等编）
- 13. 接口技术（严允中等编）
- 14. 计算机外部设备（章振业等编）
- 15. 数据库系统原理（李建中等编）
- 16. 计算机图形学（任爱华编）
- 17. Petri网原理（袁崇义编）
- 18. 《算法与数据结构》学习指导与习题解析（王晓东等编著）

电子工业出版社近期推出部分
高等学校计算机专业教材

•为电子工业部全国计算机专业教学指导
委员会确定的“九五”规划教材。

ISBN 7-5053-5725-5



9 787505 357259 >
www.aibbt.com 让未来触手可及



本书贴有激光防伪标志。凡没
有防伪标志者，属盗版图书。

ISBN 7-5053-5725-5

G·489 定价：28.00元

高等学校规划教材

人工智能基础

邵军力 张 景 魏长华 编著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本教材为中国计算机学会教育专业委员会和全国高等学校计算机教育研究会组织编写推荐出版的《计算机学科教育计划 1993》的配套教材之一，并被纳入电子工业部《1996~2000 年全国电子信息类专业教材编审出版规划》的规划教材。参考学时为 50~70 学时，内容涉及人工智能概况、知识表示、问题求解、机器学习、知识发现等机理，并介绍了人工智能发展的重要进展，等等。可作为本科生教材，也可作为研究生和在职专业人员学习教材。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，翻版必究。

图书在版编目(CIP)数据

人工智能基础/邵军力等编著. - 北京: 电子工业出版社, 2000.3

高等学校规划教材

ISBN 7-5053-5725-5

I . 人… II . 邵… III . 人工智能-理论-高等学校-教材 IV . TP18.

中国版本图书馆 CIP 数据核字(1999)第 70454 号

丛 书 名：高等学校规划教材

书 名：人工智能基础

编 著：邵军力 张 景 魏长华

责任编辑：赵家鹏

特约编辑：袁 英

排版制作：电子工业出版社计算机排版室

印 刷 者：北京李史山胶印厂

装 订 者：

出版发行：电子工业出版社 URL: <http://www.phei.com.cn>

北京市海淀区万寿路 173 信箱 邮编 100036

经 销：各地新华书店

开 本：787×1092 1/16 印张：22.5 字数：568 千字

版 次：2000 年 3 月第 1 版 2000 年 3 月第 1 次印刷

书 号：ISBN 7-5053-5725-5
G·489

印 数：4000 册 定价：28.00 元

凡购买电子工业出版社的图书，如有缺页、倒页、脱页、所附磁盘或光盘有问题者，请向购买书店调换；若书店售缺，请与本社发行部联系调换。电话 68279077

出版说明

中国计算机学会教育专业委员会和全国高等学校计算机教育研究会(以下简称“两会”),为了适应培养我国 21 世纪计算机各类人材的需要,根据学科技术发展的总趋势,结合我国高等学校教育工作的现状,立足培养的学生能跟上国际计算机科学技术发展水平,于 1993 年 5 月参照 ACM 和 IEEE/CS 联合教程专题组 1990 年 12 月发表的《Computing Curricula 1991》,制定了《计算机学科教学计划 1993》,并组织编写与其配套的首批 18 种教材。现推荐给国内有关院校,作为组织教学的参考。

《计算机学科教学计划 1993》是从计算机学科的发展和社会需要出发提出的最基本的公共要求,不是针对某一具体专业(如计算机软件或计算机及应用专业),因此它适用于不同类型的学校(理科、工科及其它学科)、不同专业(计算机各专业)的本科教学。各校可以根据自己的培养目标和教学条件有选择地组织制定不同的教学计划,设置不同的课程。本教学计划的思想是将计算机学科领域的知识,分解为九个主科目(算法与数据结构、计算机体系结构、人工智能与机器人学、数据库与信息检索、人-机通信、数值与符号计算、操作系统、程序设计语言、软件方法学与工程)作为学科的公共要求;对计算机学科的教学归结为理论(数学)、抽象(实验)和设计(工程)三个过程,并强调专业教学一定要与社会需要相结合。另外,还提出了贯穿于计算机学科重复出现的十二个基本概念,在深层次上统一了计算机学科,对这些概念的理解和应用能力,是本科毕业生成为成熟的计算机学科工作者的重要标志。

为了保证这套教材的编审和出版质量,两会成立了教材编委会,制定了编写要求和编审程序。编委会对编者提出的编写大纲进行了讨论,其中一些关键性和难度较大的教材还进行了多次讨论。并且组织了部分编委对教材的质量和进度分片进行检查落实,有的教材在编审过程中召开了部分讲课教师座谈会,广泛听取意见。参加这套教材的编审者都是在该领域第一线从事教学和科研工作多年,学术水平较高,教学经验丰富,治学态度严谨的教师。这套教材的出版得到了电子工业出版社的积极支持。他们把这套教材列为出版社的重点图书出版,并制定了专门的编审出版暂行规定和出版流程,组织了专门的编辑和协调机构。

这套教材的编审出版凝聚了参加这套教材编审教师和关心这套教材的教师、参与编辑和出版工作者、以及编委会成员的汗水,他们为此作出了努力。

这套教材还得到电子工业部计算机专业教学指导委员会的支持,其中 11 本被选入 1996~2000 年全国工科电子类专业规划教材。

限于水平和经验,这套教材肯定还会有缺点和不足,希望使用教材的单位、教师和同学积极提出批评建议,共同为提高教学质量而努力。

中国计算机学会教育专业委员会
全国高等学校计算机教育研究会

教材编审委员会成员名单

主任:王义和 哈尔滨工业大学计算机系
副主任:杨文龙 北京航空航天大学计算机系(兼北京片负责人)
委员:朱家铿 东北大学计算机系(兼东北片负责人)
龚天富 电子科技大学计算机系(兼成都片负责人)
邵军力 南京通信工程学院计算机系(兼南京片负责人)
张吉锋 上海大学计算机学院(兼上海福州片负责人)
李大友 北京工业大学计算机系
袁开榜 重庆大学计算机系
王明君 电子工业出版社
朱毅 电子工业出版社(特聘)

前　　言

本教材系中国计算机学会教育专业委员会和全国高等学校计算机教育研究会组织编写并推荐出版的《计算机学科教育计划 1993》的配套教材之一,也是原电子工业部《1996~2000 年全国电子信息类专业教材编审出版规划》的规划教材。

人工智能是一门前沿和交叉学科,尽管它的形成和发展已有近 50 年的历史,然而,时至今日,离它要在机器上再现人的智能的目标还相差甚远。现今所有研究人类智能和利用计算机软、硬件实现人类某些智能的有关理论研究、实现及应用系统开发,都是人工智能学科的研究范畴。本教材编写的宗旨是:全面而简明地介绍人工智能的基础理论和基本技术,注意理论联系实际,力求反映人工智能研究的最新进展。

本教材的参考学时为 50~70 学时。全书共分十七章。第一章叙述人工智能概况,总体介绍人工智能的研究目标、研究领域和研究途径,并简要回顾人工智能的发展历程。第二~七章介绍知识的表示方法,问题求解的基本原理及高级求解方法。第八~九章介绍机器学习、知识发现和知识积累的机理,并展现了这一领域的最新面貌。第十~十三章介绍人工智能中较为实用、并已取得重要进展的分支,即作为机器感知领域研究的自然语言理解和计算机视觉;作为知识工程领域研究的专家系统;作为进化计算领域研究的遗传算法。第十四章是人工智能语言。第十五章简要介绍与逻辑学派竞争的实现人工智能的另一重要途径,即仿生学派的人工神经网络原理。第十六章运用折衷概念研究在智能系统中如何吸收不同人工智能学派的成果,构造综合的智能系统。第十七章简要介绍现今人工智能的主要学派及他们在方法论方面的区别,以及与人工智能技术相关的研究课题。

本教材第二、五、九、十二、十七章由南京通讯工程学院邵军力教授编写,第三、四、十、十一、十四、十五、十六章由西安理工大学张景教授编写,第一、六、八章由华中师范大学魏长华副教授编写,第七章由西安理工大学李长河副教授编写,第十三章由西安理工大学周世生副教授编写,全书由邵军力教授统稿。

本教材是为本科生教学编写的,也可作为研究生和在职专业人员学习用的教材。在教材编写过程中,尽量体现了作者对《计算机学科教学计划 1993》的理解和体会。

本教材的主审为贺贯中副教授,责任编委为李伯成教授。

在本教材编写过程中,得到了杨文龙教授的多次具体指导及赵家鹏同志的关怀;陆雪莹、范建华、周兰生等博士也给予了热情的帮助,作者借此谨表诚挚的谢意。作者还要感谢贺雅娟教授给本书提出的许多宝贵意见和为本书所做的大量具体工作。

由于作者水平所限,书中难免还存在一些缺点和错误,殷切希望广大读者批评指正。

邵军力
1999 年 5 月

目 录

第一章 人工智能概述	(1)
第一节 人工智能概况.....	(1)
一、什么是人工智能	(1)
二、什么是智能	(1)
第二节 人工智能的研究途径	(3)
第三节 人工智能的研究目标	(4)
第四节 人工智能的研究领域	(5)
一、模式识别(Pattern Recognition)	(5)
二、问题求解(Problem Solving)	(7)
三、自然语言理解(Natural Language Understanding).....	(8)
四、自动定理证明(Automatic Theorem Proving)	(8)
五、机器视觉(Machine Vision)	(9)
六、自动程序设计(Automatic Programming)	(10)
七、专家系统(Expert System)	(11)
八、机器学习(Machine Learning)	(12)
九、机器人(Robots)	(12)
第五节 人工智能研究的历史回顾及进展	(12)
一、人工智能研究的历史回顾	(12)
二、人工智能研究的进展	(18)
习题一	(20)
第二章 问题求解的基本原理	(21)
第一节 状态空间法问题求解	(21)
一、问题的状态空间表示	(21)
二、状态空间的穷搜索法	(22)
三、启发式搜索法	(24)
第二节 问题归约法	(27)
一、问题归约描述	(27)
二、与或图表示	(28)
三、AO*算法	(29)
第三节 博弈树搜索	(33)
一、极大极小过程	(34)
二、 $\alpha - \beta$ 过程	(35)
习题二	(37)
第三章 基于逻辑的问题求解方法	(39)
第一节 一阶谓词逻辑基础	(39)
一、谓词逻辑的符号体系	(39)

二、谓词演算公式	(40)
三、谓词公式的解释	(40)
四、谓词演算的基本等价式及推理规则	(41)
五、谓词逻辑的演绎推理方法	(42)
六、谓词公式的规范化	(43)
七、置换与合一	(45)
第二节 基于一阶谓词逻辑的知识表达	(46)
第三节 归结(消解)原理	(46)
一、归结原理简介	(46)
二、归结原理在问题求解中的应用举例	(48)
第四节 基于规则的演绎推理	(49)
一、基于规则的正向演绎推理(FR)	(49)
二、基于规则的反向演绎推理(BR)	(50)
第五节 时序逻辑	(51)
一、时序逻辑概述	(51)
二、基于系统状态的时序逻辑	(53)
三、基于时序规则的控制领域知识表达——时序规则法	(55)
习题三	(57)
第四章 产生式系统	(59)
第一节 产生式系统概述	(59)
第二节 产生式系统的工作周期	(60)
第三节 产生式系统的控制策略	(62)
一、正向推理	(62)
二、反向推理	(65)
三、正反向混合推理	(68)
第四节 典型的产生式系统 CLIPS	(68)
一、CLIPS 的基本组成与知识表示	(69)
二、CLIPS 的推理机制	(70)
第五节 对产生式系统的评价	(73)
一、RETE 算法	(74)
二、知识库编译	(74)
三、并行处理	(75)
习题四	(75)
第五章 基于结构化表示的问题求解	(76)
第一节 语义网络	(76)
一、语义网络的知识表示	(76)
二、语义网络的推理	(81)
三、语义网络表示的特点与不足	(83)
第二节 框架系统	(83)
一、框架的构成	(84)
二、框架系统的推理	(86)
三、框架表示的特点与不足	(86)

第三节 面向对象的表示方法	(87)
一、关于对象的定义	(87)
二、消息、接口和方法	(88)
三、类(class)	(88)
四、封装与继承	(90)
五、面向对象知识表示与语义网络、框架系统的比较	(92)
习题五	(93)
第六章 不确定知识表示及推理	(94)
第一节 不确定推理概述	(94)
一、不确定性问题的代数模型	(94)
二、几种主要的不确定性推理方法	(96)
第二节 不确定推理方法	(97)
一、确定性因子法	(97)
二、主观 Bayes 方法	(101)
三、D-S 证据理论	(107)
四、可能性理论	(114)
五、信念网络	(121)
第三节 非单调推理	(122)
一、缺省推理	(123)
二、自认识逻辑	(125)
三、界限理论	(126)
四、正确性维持系统	(128)
第四节 粗集理论	(130)
一、RST 的概述	(130)
二、粗集理论的不确定性知识表示	(131)
习题六	(132)
第七章 规划求解系统	(136)
第一节 规划	(136)
一、规划的概念	(136)
二、规划的特性和作用	(136)
三、系统规划求解的方法与途径	(138)
第二节 机器规划成功性基本原理	(138)
一、概述	(138)
二、总规划的设计与分层规划原理	(139)
三、规划问题求解与最优规划原理	(140)
四、规划的双序求解与诊断	(141)
第三节 规划搜索求解	(141)
一、搜索域的分层规划	(141)
二、分层规划搜索的并行处理	(142)
三、一个实例——魔方问题的规划搜索求解	(143)
第四节 机器人规划问题求解	(143)
一、机器人工作规划及生成方法	(144)
二、规划的执行与操作控制	(144)

三、机器人过程控制与环境约束	(145)
第五节 基于谓词逻辑的机器规划设计	(145)
一、机器人行动规划问题分析	(145)
二、机器人行动规划状态与操作设计	(146)
三、机器人行动规划过程设计	(148)
四、机器人多级规划	(149)
习题七	(150)
第八章 机器学习	(152)
第一节 机器学习的概念	(152)
一、什么是学习	(152)
二、人类学习与机器学习	(152)
第二节 机器学习系统	(153)
一、什么是机器学习系统	(153)
二、一种机器学习系统模型	(154)
第三节 机器学习分类	(155)
一、基于推理策略的分类	(156)
二、基于系统性的分类	(160)
第四节 机器学习的发展简史	(161)
第五节 从例子中学习	(162)
一、概述	(162)
二、从例子学习的两个空间模型	(162)
三、学习单个概念	(167)
四、学习多个概念	(175)
五、学习执行多步任务	(179)
第六节 基于解释的学习	(180)
一、基于解释学习系统的结构	(181)
二、基于解释学习的工作原理	(181)
三、基于解释学习的方法	(181)
第七节 从观察中学习	(183)
习题八	(192)
第九章 数据库中的知识发现	(193)
第一节 引言	(193)
第二节 KDD 研究现状	(194)
第三节 KDD 的一般机理和理论基础	(196)
一、一般机理	(196)
二、主要研究方法	(197)
三、抽取知识的类型和表示	(197)
四、知识发现状态空间	(198)
五、KDD、数据挖掘及其他相关领域之间的关系	(199)
第四节 KDD 系统的基本框架	(199)
一、KDD 系统的特点	(199)
二、KDD 系统的基本框架	(200)

第五节	数据库发现知识的方法	(201)
第六节	KDD 所面临的问题和研究方向	(202)
习题九		(202)
第十章 遗传算法		(203)
第一节	遗传算法的基本概念	(203)
第二节	简单遗传算法	(203)
第三节	图式定理	(209)
第四节	遗传算法应用中的一些基本问题	(210)
一、知识表示(编码)		(210)
二、适应度函数		(211)
三、控制参数和终止判据		(211)
第五节	高级遗传算法(RGA)	(212)
一、改进的选择方法		(212)
二、高级遗传运算和算法		(213)
三、混合遗传算法		(214)
四、并行遗传算法		(215)
第六节	遗传算法(GA)应用举例	(216)
一、GA 在 TSP(旅行商)问题求解中应用		(217)
二、GA 在电网优化规划中的应用		(217)
三、GA 在木材切割优化中的应用		(218)
习题十		(218)
第十一章 专家系统概述		(220)
第一节	专家系统简介	(220)
一、专家系统概念及发展动态		(220)
二、专家系统的概念		(220)
三、专家系统应用举例		(221)
第二节	专家系统的结构及工作原理	(223)
第三节	专家系统的开发过程	(225)
第四节	PC 计算机故障诊断指导专家系统(PCDGES)示例	(226)
习题十一		(231)
第十二章 自然语言处理		(232)
第一节	自然语言处理的一般问题	(232)
一、自然语言处理的概念及意义		(232)
二、自然语言处理的发展简史		(232)
三、自然语言处理领域中的几种思想		(234)
四、自然语言处理的层次		(235)
第二节	语法层：形式语法分析	(237)
一、转换生成语法		(237)
二、扩充转移网络		(241)
第三节	语义层：格语法	(242)
第四节	语用层：篇章算法	(243)
一、篇章结构与文本连贯性		(243)

二、框架理论	(244)
三、修辞结构理论	(246)
第五节 自然语言生成	(249)
一、自然语言生成	(249)
二、语言生成系统特点分析	(249)
三、生成系统结构	(250)
四、系统功能语法	(251)
第六节 自然语言处理系统	(254)
习题十二	(254)
第十三章 计算机视觉.....	(256)
第一节 图像的理解与分析	(256)
一、关于视觉信息的表像框架	(256)
二、边缘距离的计算	(258)
三、表面方向的确定	(260)
第二节 物体形状的描述与计算.....	(262)
一、物体形状的广义锥体表示	(263)
二、广义锥体描述的计算	(263)
第三节 机器人三维视觉	(264)
一、视觉系统设计	(264)
二、臂控立体摄像机	(265)
三、三维物体识别系统	(265)
习题十三	(266)
第十四章 人工智能语言	(267)
第一节 人工智能语言概述	(267)
第二节 Prolog 语言	(267)
一、Prolog 概述	(267)
二、PDC Prolog 简介	(268)
三、Prolog 的三种基本语句	(269)
四、PDC Prolog 程序的基本结构	(270)
五、Prolog 的基本数据结构	(271)
六、Prolog 的基本工作原理	(273)
七、Prolog 程序应用举例	(274)
第三节 LISP 语言.....	(289)
一、LISP 语言的数据结构	(290)
二、LISP 语言的程序结构与基本函数	(290)
三、LISP 语言中的递归和循环	(292)
四、LISP 在产生式系统中的应用举例	(292)
第四节 小结	(295)
习题十四	(295)
第十五章 人工神经网络	(297)
第一节 人工神经网络(ANN)概述	(297)
第二节 多层前馈神经网络(BP 网络)	(299)

一、单层感知机神经网络	(299)
二、多层感知机神经网络概述	(300)
三、多层感知机神经网络(BP网络)的学习算法——反向传播算法	(301)
四、BP网络设计中的一些问题	(302)
五、多层感知机神经网络应用举例——基于神经网络的水净化控制决策系统(WCCD)	(303)
第三节 HOPFIELD 神经网络	(306)
一、HOPFIELD 神经网络概述	(306)
二、离散型 HOP 网络的基本学习规则	(307)
三、HOP 网络的应用	(309)
第四节 海明(Hamming)神经网络	(312)
第五节 自组织特征映射神经网络(SOM)	(313)
习题十五	(315)
第十六章 融合多种智能技术的智能系统	(317)
第一节 概述	(317)
第二节 ES 与 NN 的结合	(318)
一、ES 与 NN 的结合概述	(318)
二、ES 与 NN 结合的一些方法	(319)
三、ES 与 NN 结合举例	(321)
第三节 GA 与 FZ 的集成	(326)
一、概述	(326)
二、使用 GA 设计 FZ 系统	(326)
三、使用 FZ 控制 GA 系统	(327)
第四节 GA 与 NN 的集成	(328)
第五节 GA 与 ES 的集成	(330)
一、GA 与 ES 的相互辅助	(330)
二、GA 与 ES 的协同	(332)
习题十六	(333)
第十七章 人工智能的发展	(334)
第一节 人工智能的基本问题、主要学派及其观点	(334)
一、人工智能的五大基本问题	(334)
二、人工智能的主要学派	(335)
第二节 人工智能研究的策略	(336)
第三节 人工智能研究的新课题	(337)
参考文献	(339)
后记	(344)

第一章 人工智能概述

从 1946 年美国数学家麦卡锡(J·W·McCarthy)和研究生埃克特(J·D·Eckert)合作,研制成功世界上第一台通用电子数字计算机 ENIAC 以来,计算机作为一门学科得到了迅速发展。其理论研究和实际应用的深度和广度,是其他学科所无法比拟的。可以说,计算机的诞生和发展是本世纪科学技术最伟大的成就之一,对推动科学技术和社会的进步起到了巨大的作用。

探索能够计算、推理和思维的智能机器,是人们多年梦寐以求的理想。由于在理论上控制论、信息论、系统论、计算机科学、神经生理学、心理学、数学和哲学等多学科的发展和互相渗透,在技术上电子数字计算机的出现、发展和广泛的应用,人工智能(Artificial Intelligence,简称为 AI)的研究应运而生了。人工智能作为一门正在发展的综合性边缘学科,与原子能技术和空间技术一起被称为本世纪的三大科学技术成就。五十多年来,人工智能的理论研究和实际应用均得到迅速的发展。

第一节 人工智能概况

一、什么是人工智能

要想给人工智能作出一个如同数学定义那样严格的科学定义是件困难的事情。直到现在,到底“什么是人工智能?”仍然是学术界争论不休的问题,还没有一个被一致接受的论述。尽管如此,我们还是从不同的角度向读者介绍几种有影响的说法。

斯坦福大学人工智能研究中心的尼尔逊(N·J·Nilsson)教授从处理的对象出发,认为“人工智能是关于知识的科学,即怎样表示知识、怎样获取知识和怎样使用知识的科学”。麻省理工学院温斯顿(P·H·Winston)教授则认为“人工智能就是研究如何使计算机去做过去只有人才能做的富有智能的工作”。斯坦福大学费根鲍姆(E·A·Feigenbaum)教授从知识工程的角度出发,认为“人工智能是一个知识信息处理系统”。

总之,人工智能是一门综合性的边缘学科。它借助于计算机建造智能系统,完成诸如模式识别、自然语言理解、程序自动设计、定理自动证明、机器人、专家系统等智能活动。它的最终目标是构造智能机。

二、什么是智能

什么是智能?什么样的行为称得上是智能行为?这两个问题如同“什么是人工智能?”一样,至今在学术界仍然没有达到共识。

具有权威性的字典对智能给出如下的定义:

1. 通过适当的行为调整,成功地满足各种新的状况的能力;
2. 以导致所希望目标的方式来理解现有事实间的相互关系。

第一种定义反映了智能的学习能力,第二种定义描述了智能的面向目标、问题求解和理解等几方面的属性。

通常,人们认为智能是在客观世界中解决实际问题的能力,而具备这种能力至少需要下面几个方面的知识。

1. 关于客观世界中的诸多背景知识,包括历史资料和现实状况;
2. 能对所掌握的知识进行分析、选择、归纳和总结的知识;
3. 解决问题所需的策略、决策和预测的知识;
4. 问题本身所包含的专门知识。

如果计算机系统掌握了上述知识,具有一定解决问题的能力之后,就可认为该系统具有智能了。

为现代人工智能产生作出卓越贡献的英国天才数学家图灵(A·M·Turing)1950年提出了著名的图灵试验,对智能标准作了明确的定义。

图灵试验由计算机、被测试的人和主持试验的人组成。计算机和被测试的人分别在两个不同的房间内。测试过程由主持人提出问题,由计算机和被测人分别回答。被测人回答问题时尽可能地表明他是“真正的”人,计算机也尽可能逼真地模仿人的思维方式和思维过程。如果试验主持人听取对问题的回答后,分辨不清哪个是人回答的,哪个是计算机回答的,则可以认为被测试计算机是有智能的。

图灵试验虽然形象描绘了计算机智能和人类智能的模拟关系,但是图灵试验至少有下面三个问题是值得商榷的:

1. 试验主持人提出问题的标准,在试验中没有明确给出。也就是说,试验没有给出应该用什么样的问题进行提问,才能说明机器具有智能。
2. 被测人本身所具有的智力问题,图灵试验也疏忽了。试验中没有说明被测人是正常人还是智力残缺的人;是成年人还是小孩;是受过教育的人还是文盲。
3. 图灵试验仅强调试验的结果,而没有反映智能所具有的思维过程。

李耐特(D·B·Lenat)和费根鲍姆在1991年发表的论文《知识阙值理论》(On the thresholds of knowledge)中明确给出了智能的定义。他们认为智能是在一个巨大的搜索空间中寻找满意解的能力,这个巨大的搜索空间是由知识库所定义的知识空间。

李耐特和费根鲍姆认为智能不能离开知识,有了知识才谈得上智能。另外,他们还认为知识库中需要采取显式方式表示知识,只有显式表示的知识才能在知识库中发挥效力。这一论述区分了智能和本能(条件反射和非条件反射)的差别,他们认为智能活动需要通过对知识进行搜索,寻求满意解来得以实现,它并不是简单的刺激反应过程。

1976年尼尔逊提出了物理符号系统假设。假设指出所有的智能行为都等价于一物理符号系统。此物理符号系统由一个符号集组成,每个符号都是一个物理实体,若干符号可以按某种物理方法关联起来形成一个符号结构。另外,系统还含有一组作用在符号结构上以便生成其他符号结构的过程。

由此可见,尼尔逊的物理符号系统假设所描述的智能就是一个生成某些符号结构的自动机。它包括建立过程、修改过程、复制过程和消除过程等功能。

日本著名人工智能专家渡边慧教授认为:人类智能主要体现在演绎能力和归纳能力。如果

计算机具有这种能力就是有智能了。

目前关于什么是智能,什么是智能活动,仍然处于“各抒己见、众说纷纭”的阶段,人们还不能够准确地揭示其本质。这也许正是人工智能工作者锲而不舍的一个重要原因。

我们认为:如果计算机系统具有学习能力,能够对某领域的有关问题给出正确的结论或者有用的建议,而其中所使用的手段和方法(诸如学习、发现、推理、决策等)与人相似,并且能够解释系统的智能活动过程,那么,可以认为此计算机系统具有智能了。

第二节 人工智能的研究途径

由于对人工智能本质的不同理解,形成了人工智能多种不同的研究途径,主要是符号主义(Symbolism)和联接主义(Connectionism)途径。

以认知心理学派为代表的符号主义认为:人类智能的基本元素是符号,人类的认识过程就是一种符号处理过程,思维就是符号的计算。也就是说,人类的认识和思维都是可以形式化的。人类使用的自然语言本身就是用符号来表示的,人类的许多思维活动如决策、设计、规划、运筹、诊断都可以用自然语言来描述,因而也就可以用符号来表示了。

符号主义的理论基础是物理符号系统假设。许多成功的专家系统、自然语言理解系统都是基于这种观点研制的。

作为通用智能基础的 Soar 系统是符号主义者的杰作,它试图通过提供一个思维模拟的工具,来促进我们对人类智能的认识。Soar 系统的理论基础是智能行为的基础为物理符号系统。Soar 系统的结构是层次结构:记忆层(memory level),决策层(decision level)和目标层(goal level)。

记忆层位于系统的最底层,在这个层发生的活动就是知识的存储,符号的存取,即通过符号引出的知识是可以重新得到的。决策层实现知识的编码,并完成大部分初级操作。目标层的任务是建立目标,并通过决策序列达到目标。

人工智能的另一个研究途径是联接主义,即人工神经网络。联接主义根据对人脑的研究,认为人类智能的基本单元是神经元,人类的认知过程就是网络中大量神经元的整体活动。这种活动不是串行方式,而是以并行分布方式进行的。区别于符号主义,人工神经网络中不存在符号的运算。

人工神经网络的研究可以追溯到 40 年代,1943 年美国心理学家 Mceulloch 和数理逻辑学家 Pittis 在《数学生物物理》杂志上发表了一篇有关神经网络数学模型的文章。该模型被称为 M-P 模型。

1949 年心理学家海勃(D·D·Hebb)提出了突触联接强度可变的假设。假设认为:学习过程最终发生在神经元之间的突触部位,突触的联接强度随突触前后神经元的活动而变化。即若神经元 a 接收另一神经元 b 来的输入,那么当这两个神经元都猛烈活动时,从 a 到 b 的联结权就增大。此假设被称为 Hebb 学习律,它为神经网络的学习算法奠定了基础。

50 年代及 60 年代初期,一群研究人员结合生物学和心理学研究成果,开发出一批神经网络,开始用电子线路实现,后来较多的是用更灵活的计算机进行模拟。如 1957 年罗圣勃莱特(F·Rosenblatt)的感知机。它第一次将神经网络研究从纯理论付诸于物理实施。1960 年威丘(B·Widrow)提出了自适应线性元件的概念。这是一个连续取值的线性网络,用于自适应系统。

神经网络研究高潮的标志是美国加州工学院的物理学家霍普菲尔特(J·Hopfield)于1982年和1984年发表的两篇文章。文章提出的人工神经网络模型被称为 Hopfield 神经网络。

1984年黑顿(J·Hitton)等人提出了一种可行的算法,称为玻兹曼(Boltzmann)模型。由于 Boltzmann 机的结构非常简单,容易硬件化,人们利用模拟电路的噪声来模拟退火,引入一个全局性的随机参数,可以使网络的状态变量从局部极小过渡到全局的稳定点。

1986 年拉孟哈特(D·E·Rumelhart)和麦克里兰德(J·L·McClelland)提出了 PDP(Parallel Distributed Processing)理论,试图探讨人类认识过程的微观结构,同时也提出了多层神经网络的误差反传(BP)算法。基于 BP 算法的多层神经网络是当代应用最为广泛的神经网络之一。

1988 年美国加州大学蔡少堂(L·O·Chua)等人提出了细胞神经网络模型。这是一个大规模非线性模拟系统,同时具有细胞自动机的动力特征。此模型用于图像处理取得了良好的效果。

应该指出的是,虽然经过众多科学家坚持不懈的努力,在神经网络研究中取得了大量成果,但是由于神经网络研究的复杂性,目前还是处于基础性的研究阶段,还有待于数学家、物理学家、微电子学家、生物学家、心理学家、控制理论学家和计算机科学家们共同努力,进行更加艰苦卓越的研究工作。

第三节 人工智能的研究目标

人工智能的研究目标可划分为近期目标和远期目标两个阶段。

人工智能近期目标的中心任务是研究如何使计算机去做那些过去只有靠人的智力才能完成的工作。根据这个近期目标,人工智能作为计算机科学的一个重要学科,主要研究依赖于现有计算机去模拟人类某些智力行为的基本理论、基本技术和基本方法。五十多年来,虽然人工智能在理论探讨和实际应用上都取得了不少成果,但是仍有不尽人意之处。尽管在发展的过程中,人工智能受到过重重阻力,而且曾陷于困境,但它仍然在艰难地向前发展着。

探讨智能的基本机理,研究如何利用自动机去模拟人的某些思维过程和智能行为,最终造出智能机器,这可以作为人工智能的远期目标。

这里所说的自动机并非常规的计算机。因为现有常规计算机属冯·诺依曼(J. Von Neumann)体系结构,它的出现并非为人工智能而设计。常规计算机以处理数据世界中的问题为对象,而人工智能所面临的是事实世界和知识世界。智能机器将以事实世界和知识世界的问题求解为目标,面向它本身处理的对象和对象的处理过程而重新构造。人工智能研究的远期目标的实体是智能机器,这种机器能够在现实世界中模拟人类的思维行为,高效率地解决问题。

从研究的内容出发,李艾特和费根鲍姆提出了人工智能的九个最终目标:

1. 理解人类的认识 此目标研究人如何进行思维,而不是研究机器如何工作。要尽量深入了解人的记忆、问题求解能力、学习的能力和一般的决策等过程。

2. 有效的自动化 此目标是在需要智能的各种任务上用机器取代人,其结果是要建造执行起来和人一样好的程序。

3. 有效的智能拓展 此目标是建造思维上的弥补物,有助于使我们的思维更富有成效、更快、更深刻、更清晰。

4. 超人的智力 此目标是建造超过人的性能的程序。如果越过这一知识阈值,就可以导

致进一步的增殖,如制造行业上的革新、理论上的突破、超人的教师和非凡的研究人员等。

5. 通用问题求解 此目标的研究可以使程序能够解决或至少能够尝试其范围之外的一系列问题,包括过去从未听说过的领域。

6. 连贯性交谈 此目标类似于图灵测试,它可以令人满意地与人交谈。交谈使用完整的句子,而句子是用某一种人类的语言。

7. 自治 此目标是一系统,它能够主动地在现实世界中完成任务。它与下列情况形成对比:仅在某一抽象的空间作规划,在一个模拟世界中执行,建议人去做某种事情。该目标的思想是:现实世界永远比我们的模型要复杂得多,因此它才成为测试所谓智能程序的唯一公正的手段。

8. 学习 该目标是建造一个程序,它能够选择收集什么数据和如何收集数据,然后再进行数据的收集工作。学习是将经验进行概括,成为有用的观念、方法、启发性知识,并能以类似方式进行推理。

9. 储存信息 此目标就是要储存大量的知识,系统要有一个类似于正文百科词典式的,包含广泛范围知识的知识库。

总之,无论是人工智能研究的近期目标,还是远期目标,摆在我们面前的任务异常艰巨,还有一段很长的路要走。在人工智能的基础理论和物理实现上,还有许多问题要解决。当然,仅仅只靠人工智能工作者是远远不行的,还应该聚集诸如心理学家、逻辑学家、数学家、哲学家、生物学家和计算机科学家等,依靠群体的共同努力,去实现人类梦想的“第二次知识革命”。

第四节. 人工智能的研究领域

在人工智能学科中,按照所研究的课题,研究的途径和采用的技术考虑,它所包括的研究领域有模式识别、问题求解、自然语言理解、自动定理证明、机器视觉、自动程序设计、专家系统、机器学习、机器人等。此处并不打算详细论述这些应用领域的相关理论和技术,而仅仅介绍这些领域研究所涉及到的一些基本概念和基本原理。目的是使读者对人工智能这门学科的研究领域能有一个总体了解,为后续章节的学习奠定基础。

一、模式识别(Pattern Recognition)

模式识别是人工智能最早研究的领域之一。它是利用计算机对物体、图像、语音、字符等信息模式进行自动识别的科学。

(一) 模式识别的过程

模式识别过程一般包括对待识别事物进行样本采集、信息的数字化、数据特征的提取、特征空间的压缩以及提供识别的准则等。此过程如图 1-1 所示,图中虚线下部是学习训练过程,上部是识别过程。

在学习的过程中,首先将已知的模式样本进行数值化,送入计算机,然后将这些数据进行分析,去掉对分类无效的或可能引起混淆的那些特征数据,尽量保留对分类判别有效的数值特征,这个过程亦称为特征选择。有时,还得采用某种变换技术,得出数量上比原来少的综合性特征(称为特征空间压缩,亦称为特征提取),然后再按设想的分类判别的数学模型进行分类,并

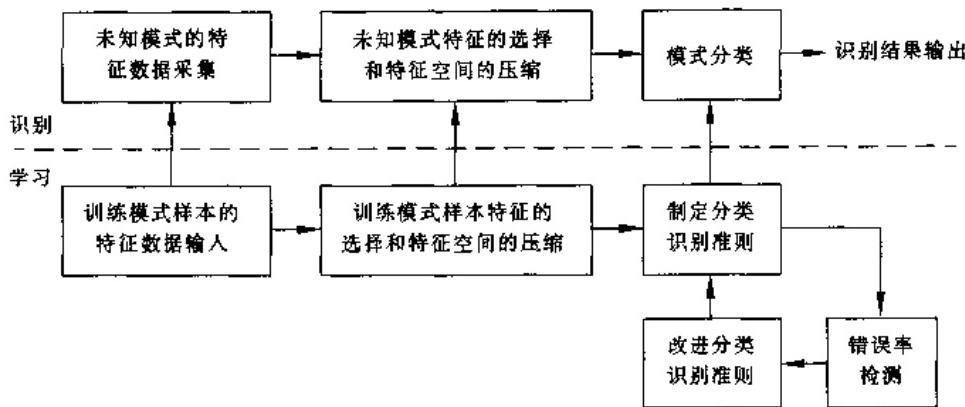


图 1-1 模式识别的过程

将分类结果与已知类别的输入模式进行对比，不断修改，制定出错误率最小的判别准则。

(二) 模式识别的分类

模式识别常用的方法有统计决策法与句法方法，监督分类与非监督分类法和参数与非参数法等。

1. 统计决策法与句法方法

统计决策法利用概率统计的方法进行模式识别。它先对已知样本模式进行学习，通过样本特征建立起判别函数，当给定某一待分类模式特征后，看它落在特征超平面上判别函数的哪一侧，就可以判别它是属于哪个类别了。

句法方法也称为结构法。它把模式分解成若干个简单元素，然后用特殊文法规则描述这些元素之间的结构关系。不同的模式对应着不同的结构。例如图 1-2 所示的图片，现在对图片的结构作一描述。这种描述可采用树结构来实现，如图 1-3 所示。事实上，这种描述使用了形式语言的技巧。

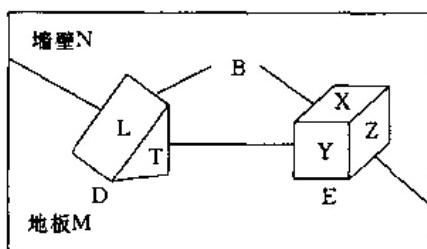


图 1-2 结构法示例

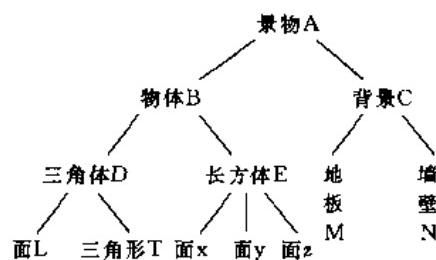


图 1-3 图片的树结构概述

句法方法适合于结构明显、噪声很少的模式识别。现已用于文字、染色体、指纹等的识别。

2. 监督分类与非监督分类

如果模式样本有 n 个特征，那么一个样本就构成了一个 n 维的特征向量，它在 n 维空间就对应一个点。所谓的分类问题就是把特征空间分割成对应于不同类别的互不相容的区域，每一个区域对应于一个特定的模式类，而不同类别之间的界面用“判别函数”来描述。

对于监督分类，需要根据样本的特征向量来确定判别函数，只有在判别函数确定之后，才能够用它对未知模式进行分类。同时，要知道待分类模式足够的先验知识。

在缺乏待分类模式的先验知识的情况下，就要采用非监督分类，即聚类分析。聚类分析方

法是用数学的方法分析各特征向量之间的距离及分散程度。有些特征向量可能聚集成若干个群，可以按各个群之间的距离远近进行分类。

聚类分析有两种基本方法，其一是分级聚类方法，其二是迭代最优化方法。无论采取哪种方法，重要的是选取合适的聚类准则和类间相似性的测度。聚类准则应是使类间相似度尽量小，而类内相似度尽量大。类间相似性的测度一般定义为两类样本之间的最小、最大距离或两类样本距离的某种统计量，有时还要考虑样本之间的近邻关系。

3. 参数与非参数法

参数法亦称为参数估计法。它是当模式样本的类概率密度函数的形式已知，或者从提供的作为设计分类器用的训练样本能估计出类概率密度函数的近似表达式的情况下使用的一种模式识别方法。例如，在多数情况下，类概率密度函数常常用正态分布来近似，即用正态分布的均值和协方差矩阵作为估计计算判别函数的参数。

参数估计中最常用的方法是最大贝叶斯估计和最大似然估计。

如果样本的数目太少，难以估计出概率密度函数，这时就要使用非参数估计法。

非参数估计方法常用的有 k -最近邻判定规则。其基本思想是直接按 k 个最近邻样本的不同类别分布，将未知类别的特征向量分类。

二、问题求解(problem solving)

人工智能中的问题求解是指通过搜索的方法寻找问题求解操作的一个合适序列，以满足问题的要求。

问题求解的基本方法有状态空间法和问题归纳法。

问题求解的状态空间法可以描述为：

若定义 S 为被求解问题可能有的初始状态的集合， F 为求解过程中可使用的操作的集合，而 G 为目标状态的集合，那么问题求解的过程则是在状态空间中寻找从初始状态 X_0 出发，到到达目标状态 X_g 的一个路径。这个路径称之为解路径。

一般情况下，问题求解程序由下面三个部分组成：

1. 数据库 数据库中包含与具体任务有关的信息，这些信息描述了问题的状态和约束条件。状态分量的选择应该满足独立性、必要性和充分性。各个分量不同的取值组合对应着不同的状态，但并不是所有的状态都是问题求解所需要的。问题本身所具有的约束条件可以帮助除去那些非法的状态和不可能出现的状态，而保留在数据库中的是问题的初始状态、目标状态和中间状态。

2. 操作规则 由于数据库中的知识是叙述性知识，而操作规则是过程性知识。系统中的操作规则都由条件和动作两部分组成，条件给定了操作的适应性的先决条件，动作描述了由于操作而引起的状态中某些分量的变化。

3. 控制策略 系统中的控制策略确定了求解过程中应该采用哪一条适用的规则，所谓的适用规则是指从规则集合中选择出最有希望导致目标状态的操作，施加到当前状态上，以便克服组合爆炸。

问题求解的状态空间法通常是一种搜索技术。基本的搜索策略有：深度优先法、广度优先法、爬山法、回溯策略、图搜索策略、启发式搜索策略，与或图搜索和博奕树搜索等。

三、自然语言理解(Natural Language Understanding)

自然语言理解包括问答系统、声音理解系统、手书文字识别系统和机器翻译系统等。

从通用计算机问世以来，人们就想用计算机把一种语言翻译成另一种语言。当初，人们认为翻译包括两个基本过程，即查词典和语法分析，但是使用这种方法并没有达到预期效果，这是因为人们在理解某种语言的时候，不仅要依赖于他的语法知识，而且还要运用他所涉及世界的有关知识。

从 1966 年以来，同自然语言理解相关的研究，几乎都转向问答系统的探讨上来，即用某种方法去研究建立一个计算机系统，让机器理解语言。该系统根据机器对人所提问的问答，就可以判断机器是否理解了人所说的语言。

自然语言理解的研究具有十分重要的意义。

1. 如果人们能用自己的语言同计算机打交道，而不必为使用计算机去学习程序设计语言，这对计算机的广泛应用无疑具有深远意义。

2. 自然语言理解的研究可以更好地了解人类大脑是如何工作的。语言是人类思维不可分割的一部分，人类的记忆、推理、意识都是与语言是如何工作的这个问题密切相联。

3. 人工智能工作者在自然语言理解的研究过程中，注意力集中在语言的功能上，即把语言看作是一个智能生物同另一个智能生物的通信过程。书写程序是为了在计算机上完成专门的语言任务，在书写和利用这种程序在计算机上实验的过程中，人们可能形成和发展人类语言处理的理论基础的概念和技术。

1970 年 Winograd 在计算机上实现了一个灵活地结合句法、语义、推理以及上下文和世界知识的程序，成功地进行人—机间的对话，这是 Winograd 对自然语言理解所作的卓越贡献。他的贡献在于要使机器理解语言，不仅要考虑句法，还要考虑语义，利用知识，要引进一般的社会知识，以及利用上下文信息。现在，人们认识到把所有语法、语义、语用(pragmatics)等诸因素结合起来，对句子进行适当的分析和解释的重要性。

近年来，从知识表示方面去研究自然语言理解取得了进展。如框架(frame)理论可以解决世界知识的组织问题；剧本(script)对理解报纸、故事方面发挥作用。

另外，利用机器归纳推理和抽象化处理，人们采用“檄文摘”的方式来实现自然语言理解已成为现在研究的方向。

四、自动定理证明(Automatic Theorem Proving)

自动定理证明在人工智能的研究中是一个极其重要的领域。如基于谓词演算的推理自动化研究，使我们更清楚地理解某些推理的细节。许多非数学领域的问题，如医疗诊断、信息检索、规划制定和难题求解，都可以像定理证明问题那样进行形式化，从而转化为一个定理证明问题。因此，自动定理证明在人工智能研究中起着重要作用。

自动定理证明的方法通常有：

1. 自动演绎法 自动演绎法是自动定理证明最早使用的一种方法。1956 年纽厄尔、肖和西蒙的逻辑理论家(LT)程序和 1959 年吉勒洛特(Gelernter)等人的几何定理证明机(Geometry Theorem-Proving Machine, 简称 GTM)就使用这种方法。

LT 采用“正向链”推理方法，其基本思想是依据推理规则，从前提出发并依靠公理向后推理，可得出许多定理，如果待证明的定理恰在其中，则定理得证。而 GTM 采用的是“反向链”推理方法，其基本思想是从目标出发向前提推导，依靠公理产生新的子目标，这些子目标逻辑蕴涵着最终目标。

2. 决策过程法 所谓的决策过程是指判断一个理论中某个公式的有效性(validity)。例如，1980 年依沃(Eevo)等人提出了使用集合论的决策过程；1980 年尼尔逊等人提出了带有不解释函数符号的等式理论决策过程；1978 年我国著名的数学家、计算机科学家吴文俊教授提出了关于平面几何和微分几何定理的机器证明方法。吴文俊教授方法的基本思想是：首先将几何问题代数化，即通过引入坐标，把有关的假设和求证部分用代数关系表述，然后再处理表示代数关系的多项式，即把判定多项式中的坐标逐个消去，采用多项式的消元法来验证，如果消去后结果为零，那么定理得证；否则再进一步检查。这个算法已在计算机上证明了不少难度相当高的几何问题，得到国际学术界的赞扬，被认为是当时定理机器证明和决策中最好的一种方法。

3. 定理证明器 定理证明器是研究一切可判定问题的证明方法。它的基础是 1964 年鲁宾逊(J·A·Robinson)提出的归结原理(Resolution Principle)。用归结原理所形式化的逻辑里，没有公理，只有一条使用合一(unification)替换的推导规则，这样一个简捷的逻辑系统，却是谓词演算中的一个完备系统。也就是说，任意一个恒真的一阶公式，在鲁宾逊的逻辑系统中都是可证的。

归结原理的确十分诱人，它使得许多研究者投入到对归结原理的改进工作中。重要的改进工作有：

- (1) 1964 年沃斯(Wos)，卡逊(Carson)和鲁宾逊(Robinson)提出了单文字子句优先策略和支撑集策略；
- (2) 1965 年鲁宾逊提出超归结方法(Hyper-resolution)；
- (3) 1967 年斯拉格尔(J·R·Slagle)提出语言归结；
- (4) 1968 年努夫兰德(Loveland)和拉克哈孟(Luckham)提出线性归结；
- (5) 1970 年波叶(R·Boyce)提出锁归结；
- (6) 1968 年沃斯和鲁宾逊提出了一种和归结方法相容的处理等式的方法，即调解方法(paramodulation)。

1972 年波叶和孟尔(J·Moore)开始着手研制与沃斯等基于归结方法的证明器不同的证明器。他们研制的证明器是基于类人方法的证明器。这个证明器集中于归纳领域，使用一阶逻辑语言，将归结原理与重写规则结合在一起，把已证的等式，按用户的选择转变为重写规则，以便以后使用。该系统的重要功能是能够提出归纳方案，然后对归纳方案进行分析、比较、整理，得出对所证问题适用的归纳方案。

五、机器视觉(Machine Vision)

机器视觉研究的任务是理解一个图像，即利用像素(pixels)所描绘的景物。其研究领域涉及到图片处理、图像处理、模式识别、景物分析、图像解释、光学信息处理、视频信号处理以及图像理解。这些领域可分成信号处理、分类和理解三类。

1. 信号处理 信号处理研究把一个图像转换为具有所需特征的另一个图像的方法。比

如,人们往往想要使所输出的图像尽可能具有较好的信号—干扰比,或者使图像的某些特征得到增强,以便于人们观察。这种处理技术通常称为图像处理或图片处理。数字技术、光学技术和电气的视频信号处理技术通常是信号处理中所采用的技术。

2. 分类 分类技术研究如何把图像划分为预定类别。分类是从图像中抽取一组预先确定的特征值,然后根据用于多维特征空间的统计决策方法决定一个图像是否“符合”某一类。这类方法一般称为模式识别或模式分类。

5. 理解 在给定某一图像的情况下,一个图像理解程序不仅描述这个图像的本身,而且也描述该图像所描绘的景物。对于一个图像的理解,需要任务领域的先验知识和复杂的图像处理技术。

视觉对于人来说是很容易的,但要构成一个可以与人相比的计算机视觉系统是非常困难的。这些困难也正是机器视觉所研究的课题。

(1)因为一个图像不能提供足够的信息恢复其具有的景物。一个三维的景物通过投影,变成一个二维图片,它的深度方向被折叠起来了。为了解决这些不确定性需要附加的限定,这些限定基于合理的假定和测量,如果没有这些限定,视觉工作将无法进行。

(2)一个物体的外观受到其表面材料、环境条件、光源角度、周围的光线、照相机的角度和特性的影响,造成了在一个图像中许多因素混淆在一起,很难确定每种因素对某一具体像素所起的作用。

(3)领域先验知识对图像理解所起的作用不能低估。然而,这些先验知识在所观察的图像中又往往非常弱,这就造成了理解图像的难度。

(4)人类虽然是视觉专家,但要弄清楚人们是怎样去看的却是一个困难的问题。也就是说,很难用问题求解那样的方法,形成一个视觉的分析方法。

(5)计算机视觉系统必须处理大量的信息,造成了工程上的实际困难。例如,在空中拍摄一张照片,经数字化后有 3000×3000 个像素,每个像素 8 位,每个图像就有 9 兆字节。如果每个像素要进行 10 次操作以便进行边缘检测,那么就要对一个图像进行 9 千万次操作,可见其工作量之巨大。

上述问题本身就是多年对视觉研究的成果,随着这些问题的进一步解决,机器视觉系统的能力就可得到进一步的增强。

六、自动程序设计(Automatic Programming)

编制和调试一个复杂的计算机程序是件费时的繁琐工作。一方面,具有错误的程序比比皆是,而完美、无懈可击的程序却极少有;另一方面,计算机不允许程序存在错误,程序的失误带来的后果是极其严重的,有时甚至是不能容忍的,这就造成了程序设计的困境。为了摆脱这种状况,就要从软件开发技术方面寻找出路。可以说,自动程序设计是从人工智能方面解决此问题的一种方法。

自动程序设计主要关心下面两个问题:

(一)程序验证(Program Verification)

程序验证是利用一个已验证过的程序系统来自动证明某一给定程序 P 的正确性。

设程序 P 的输入是 x ,它必满足输入条件 $\varphi(x)$;程序的输出是 $Z=P(x)$,它必满足输出条

件 $\Psi(x, z)$, 则这时程序的正确性有三种类型:

1. 部分正确性 若对任意输入 x , 只要 $\varphi(x)$ 为真且程序终止, 则 $\Psi(x, P(x))$ 为真, 称程序 P 是部分正确的。
2. 终止性 若对任意输入 x , 只要 $\varphi(x)$ 为真, 则程序 P 终止。称程序 P 是终止的。
3. 完全正确性 若对任意输入 x , 只要 $\varphi(x)$ 为真, 则程序 P 终止, 且 $\Psi(x, P(x))$ 为真。程序 P 完全正确。

(二) 程序综合(Program Synthesis)

程序综合指根据所给定问题的具体描述由计算机自动生成满足要求的程序 P 。根据这个思想, 所生成的程序 P 应对输入条件 $\varphi(x)$ 和输出条件 $\Psi(x, z)$ 是完全正确的。目前程序综合的基本途径主要是程序变换, 即通过对给定的输入、输出条件进行逐步变换, 以构成所要求的程序。

七、专家系统(Expert System)

专家系统是当前人工智能应用中最成功的一个领域。

从 1965 年费根鲍姆研制第一个专家系统 DENDRAL 以来, 专家系统以它所产生的巨大经济效益和社会效益, 已扩展到数学、物理、化学、医学、地质、气象、农业、法律、教育、交通运输、机械、艺术以及计算机科学本身等领域, 甚至还渗透到政治、经济、军事等重要决策部门。

专家系统发展之所以如此迅速, 主要因为:

- (1) 专家系统解决实际问题的周密性;
- (2) 人类专家知识的系统组织, 对其应用领域的发展起到了促进作用;
- (3) 专家系统突出知识的价值。通常推广和应用专家的知识, 要通过培训的方法, 这需要很长的时间, 而专家系统大大减少了知识传授和应用的代价, 使其可以获取很大的经济效益。

专家系统具有下面三个特性:

- (1) 启发性 它运用规范的专门知识和直觉的评判知识进行问题求解;
- (2) 透明性 它使用户能够在无需了解其系统结构的情况下与专家系统直接交往, 了解其知识内容和推理过程;
- (3) 灵活性 它可以不断接收新知识, 调整有关的控制信息, 使其与整个知识库协调。

下面简略介绍几个著名的专家系统。

DENDRAL 系统根据质谱仪所产生的数据, 不仅可以推断出确定的分子结构, 而且还可以说明未知分子的谱分析。据说该系统已经达到化学博士的水平。

MYCIN 是第一个功能较全的医疗诊断专家系统。它是 1974 年由斯坦福大学 HPP 科研小组研制成功的。该系统在不知道原始病原体的情况下, 判断如何用抗菌素来处理血液细菌感染病患者。输入系统的原始数据是患者的症状、病史和化验结果, 系统可以根据专家知识和输入的资料判断出是什么病菌引起的感染, 然后提出治疗的方案。

PROSPECTOR 是斯坦福国际研究所计算机科学技术部人工智能中心的杜达(R·O·Dude)、哈特(P·E·Hart)等人研制的一个地质勘探专家系统, 具有以下三个功能:

- (1) 勘探评价 地质工作者在某一地区获得一些有用资料后, 系统可以提供帮助。即系统可以对这些资料进行分析和评价, 预测成矿的可能性, 并指导用户下一步应采集哪些资料对勘

探工作者有价值。

(2)区域资源评价 系统处理某区域的地质数据,其处理结果是这个区域地质矿藏资源的分布,可用来对该地区的资源进行评估。

(3)井位选择 当确定某一地区蕴藏某种矿后,系统可以帮助地质工作者选择最优钻井位置,以避免不必要的浪费。

八、机器学习(Machine Learning)

自从 1980 年在卡内基-梅隆大学(Carnegie-Mellon University)召开第一届机器学术研讨会以来,机器学习的研究工作发展很快,已成为人工智能学科的中心课题之一。

目前,机器学习领域的研究工作主要围绕以下三个方面进行:

(1)面向任务的研究 研究和分析改进一组预定任务的执行性能的学习系统。

(2)认知模型 研究人类学习过程并进行计算机模拟。

(3)理论性分析 从理论上探索各种可能的学习方法和独立于应用领域的算法。

近年来,在基于解释的学习、概念形式化模型、归纳概念获取、从数据库自动发现规律、机器学习工具、诊断专家系统的错误和新增加规则等领域异常活跃,这将对产生新的计算机体系结构提供一个良好的背景。

九、机器人(Robots)

我们可以把机器人定义为一种可再编程序的多功能操作装置。机器人可以代替人从事有害环境中的危险工作,从而提高工作质量和生产效率,降低成本。在科学研究上机器人为人工智能提供了一个综合试验场所,它可以全面地检查人工智能各个领域的技术,并探索这些技术之间的关系。

就机器智能来考虑,机器人可以行使下面的功能:

(1)模式识别 给机器人配备视觉、触觉,使其能够识别可见景物的实体和阴影,甚至可以辨别出两幅图像间的微小差别,从而完成模式识别的功能。

(3)运动协调推理 机器人的运动协调推理功能是依赖感觉驱动的。感觉是机器人接受外界的刺激,而运动就是机器人的行动。这两者之间的关系实际上构成一个产生式系统。

机器人和机器人学(robotics)的研究促进了人工智能思想的发展。它所导致的一些技术可以用来模拟物理世界,也可用来描述从一种物理状态转变为另一种物理状态的过程。

第五节 人工智能研究的历史回顾及进展

一、人工智能研究的历史回顾

本世纪 40 年代,一批科学家认真总结前人在哲学、心理学、逻辑学、数学和物理学等学科的成果,开创自己富有创造性的工作。这些工作为以后的人工智能和计算机科学的奠基产生了深远的影响。

这个时期几项卓越的工作是：

1. 麦克卡洛克(W·S·McCulloch)和比脱斯(W·Pitts)神经逻辑计算的研究。1943年心理学家麦克卡洛克和数学家比脱斯从信息处理的观点,采用数理模型的方法对神经细胞动作进行研究,提出了二值神经元阈值模型。这项工作被后人公认为是对神经网络计算认真研究的开始。

2. 维纳(N·Wiener)的控制论。维纳控制论的学术思想可以说是历史上对机器智能在哲学、理论和方法上的一次全面讨论。遗憾的是,维纳控制论思想所基于的最优预测理论要求被求解的问题满足严格的统计性质。

3. 图灵的理想计算机模型 1936年英国数学家图灵提出了一个理想计算机模型(即图灵机),创立了自动机理论,将“思维”机器研究和计算机理论研究向前推进了一大步。图灵的贡献主要表现在：

- (1)他提出了基于离散量的递归函数作为智能描述的数学基础;
- (2)给出了基于行为主义的测试机器是否有智能的标准,即图灵试验。

4. 香农(Shannon)的信息论。美国数学家香农,1948年发表了《通讯的数学理论》,它标志着一门新学科——信息论的诞生。信息论对心理学产生了很大的影响,而心理学又是人工智能研究的重要支柱。认知心理学派从信息论那里得到了启发,认为人的心理活动可以通过信息的形式加以研究,并在此基础上提出了各种描述人的心理活动的数学模型。

这个时期的一些研究工作对人工智能的产生与发展起到了重要的作用。这些研究成果甚至对近代科学技术的影响也是非常巨大的。

真正对人工智能学科起到划时代作用的还算四十多年前的一次著名会议。1956年夏季,在美国达特玛斯(Dartmath)大学,由年轻数学助教麦卡锡(J·W·McCarthy)联合他的三个朋友明斯基(M·L·Minsky,哈佛大学年轻的数学家和神经学家)、朗切斯特(N·Lochester,IBM公司信息研究中心负责人)和香农(贝尔实验室研究信息的数学家),共同发起,邀请了IBM公司的莫尔(T·More)和塞缪尔(A·L·Samuel)、麻省理工学院的塞尔夫利奇(O·Selfridge)和索罗孟夫(R·Solomonoff)以及兰德公司(RAND)和卡纳奇(Carnagie)工科大学的纽厄尔(A·Newell)和西蒙(H·A·Simon)等人参加,一起探讨用机器模拟人类智能的问题。整个研讨会历时两个月之久,在会上,他们第一次正式使用了人工智能这一术语。这次具有历史意义的研讨会,标志着人工智能这门新兴学科的正式诞生。

人工智能一诞生,就以新兴事物的姿态崭露头角。就在这一年,人工智能在实验研究上取得了两项重大的成果。

其一是美国纽厄尔、肖和西蒙合作,编制了称为逻辑理论机的程序系统。该程序模拟了人用数理逻辑证明定理时的思维规律,它用分解(把一个问题分解为若干个子问题)、代入(用常量代入变量)和替换(用一个逻辑符号替换另一个逻辑符号)等方法,来处理待证的定理。如果这些子问题最终能变换为已知的公理或已证明过的定理形式,那么该定理就得证了。分解、代入和替换属于推理规则。先解决子问题,然后再解决总问题是程序给定的解题步骤来完成的。该程序证明了名著《数学原理》第二章中的38条定理。后来经过改进,又于1963年证明了该章中的全部52条定理。这是用计算机对人的思维活动进行研究的第一次成功的探索。

其二是塞缪尔于1956年研制的跳棋程序。该程序具有自学习、自组织和自适应能力。这是一个启发式程序,可像一名优秀棋手那样,向前看几步后再走棋。它可以向人学习下棋经验或自己积累经验,还可以学习棋谱。这是模拟人类学习过程的一次卓有成效的探索。1959年这

个程序已击败了它的设计者,1962年又击败了美国一个州的冠军。

除此之外,1956年另一个有影响的成就是乔姆斯基(N·Chomsky)提出了一种文法的数学模型,开创了形式语言的研究。形式语言和自动机是等价的,它们都可以用来研究人的思维过程。

也是在1956年,塞尔夫利奇研制出第一个字符识别程序,接着又在1959年推出功能更强的模式识别程序。

定理证明在1958年取得了新的成就。美籍数理逻辑学家王浩,在IBM 704计算机上证明了《数学原理》中有关命题演算的全部220条定理。他还证明了该书中150条谓词演算定理中的85%,时间只用了几分钟。1959年,王浩仅用了8.4分钟的时间就证明了上述全部定理。

从1957年开始,纽厄尔、肖和西蒙等人开始研究一种不依赖于具体应用领域的通用问题求解系统(General Problem Solving,简称为GPS),GPS是在LT的基础上发展起来的。系统建成后,又反过来应用于改进LT。GPS的研究前后共持续了10年,最后的版本是1969年公布于世的。

GPS使用了两个基本原理:中间一结局分析和递归问题求解。中间一结局分析是一种确信只有存在某个应用目的时才试图采用某个操作符(即算符)的技术。假定已定义了两个物体的可能特征,根据定义,两物体之间的某种差别是它们的某些特征值的一种差异。某个操作符改变这些特征值,由表列出操作符和它们引起的差别,规定了一个操作符的差别表后,再应用这张表来指导对子问题的选择。

GPS的工作基础是纽厄尔、肖和西蒙他们通过心理学实验,发现人在解题时的思维过程大致可分为以下三个阶段:

- (1)首先想出大致的解题计划;
- (2)根据记忆中的公理、定理和解题规则,按计划实施解题过程;
- (3)在实施的解题过程中,不断进行方法和目的的分析,修订解题计划。

这是一个具有普遍意义的思维活动过程,其中最活跃的是方法和目的的分析。

1960年麦卡锡研制出表处理语言LISP。可以方便地处理符号,在人工智能的各个研究领域中都得到广泛的应用。

1961年明斯基发表了题为“走向人工智能的步骤”的论文,对当时的人工智能研究起了推动作用。

1964年鲁宾逊提出了著名的归结原理,标志着人工智能中定理的机器证明(或是自动演绎)这个分支的开始。由于该原理的全部推导过程只需运用一条简单的推理规则,颇引人入胜,曾一度被誉为人工智能研究中重大方法论的突破。

到了70年代初期,人们发现归结原理并不能解决人工智能的一切问题,甚至连具有中等难度的数学定理也不能够圆满地证明,从而使得一度对归结原理的“热望”演变成为“失望”。不过人们冷静下来进行较缜密的分析后,发现企图用任何一种与问题领域无关的纯解析的全能形式系统来解决一切问题的努力均属徒劳无功。因而促使人们致力于研究开发各种非归结的、基于规则的演绎系统和自然演绎系统。

综观人工智能的发展历史,人们清楚地认识到这样的一个基本事实:自从计算机诞生之日起,它就以所特有的、基于符号处理的推算能力来区别于传统的计算,并展现出有可能实现某种机器智能的前景,这一前景鼓舞着计算机工作者去开拓人工智能的新领域。人们为这个领域所取得的每一项成果而欢欣鼓舞。然而,随着研究的深入,人们逐渐认识到这条通向智慧之门

的道路，并非处处都是阳光灿烂，对它寄予过高的希望看来是不切实际的。

纽厄尔和西蒙在 1958 年曾作出预言：

不出十年，计算机将成为世界象棋冠军，除非通过一个比赛规则不准它参加；

不出十年，计算机将发现并证明那时还未证明的数学定理；

不出十年，计算机将谱写出具有相当美学价值并被评论家们认可的乐曲；

不出十年，大多数心理学家的理论将采用计算机程序来形成。

然而，时间已过四十多年，至今他们的预言一个也没有得到完全实现，他们本人也不得不承认这个现实。西蒙教授于 1988 年 12 月应邀出席在东京举行的《第二次第五代计算机系统国际会议(FGCS'88 日本东京)》，在会上作了报告。报告中有一段耐人寻味的话：

“从一开始，人工智能和认知科学工作者就因过分地乐观而受人的指责。我希望我们已为某些乐观而感到内疚了，而且对于一个经历了 30 多年历程才走到今天这一步的一个领域来说，我也不认为这种指责和内疚是过分的。”

那么是什么原因导致这些预言不能得到实现，阻滞着人工智能的发展呢？下面我们就人工智能的几个相关问题进行一下分析。

1. 图灵机和冯·诺依曼机的启示

图灵机是按照一定算法实现符号处理的抽象自动机。机器智能依赖于实现的环境，即机器智能的物理载体，则是以“存储程序概念”为核心设计思想而构造的冯·诺依曼机。在冯·诺依曼机中，计算机推理是完全集中在处理部件(即 CPU)中，而完成计算和推理所需要的知识则在存储器中。计算机在任何一个时间片只能进行一步计算，而且每一步计算仅访问存储器中的一、二个单元。在给定的一个时间片内，机器仅能处理系统知识的一个极小部分。

人工智能研究者认为智能活动需要许多知识片间的相互密切配合作用，进行智能信息处理的计算机结构都应支持这种知识片之间的相互作用，以利于提高效率。对于每一个存储单元，不仅应成为知识信息的存储场所，而且也是知识信息处理活动的场所。现在看来，传统冯·诺依曼机无法满足上述要求。也就是说冯·诺依曼机的智能能力与人还相差甚远。人可以认识多年未见面、面貌大变的老同学、老朋友；可以很敏捷地在十分复杂的环境下对事物作出判断。若用计算机去做这些事情却是十分困难的。

2. 计算机博弈的困难

博弈是自然界的一种普遍现象。它表现在对自然界事物的对策或智力竞争上。博弈不仅存在于人们用于娱乐的下棋之中，而且存在于政治、经济、军事和生物的斗智和竞争之中。

尽管西洋跳棋和国际象棋的计算机程序已经达到了相当高的水平，然而计算机博弈依然面临着巨大的困难。这主要表现在以下两个方面的问题。

其一是组合爆炸问题。状态空间法是人工智能中最基本的形式化方法。若用博弈树来表示状态空间，对于几种常见的棋类，其状态空间都大得惊人。例如，西洋跳棋为 10^{40} ，国际象棋为 10^{120} ，围棋则是 10^{700} 。如此巨大的状态空间，现有计算机是无法忍受的。

其二是现在的博弈程序往往是针对二人对弈，棋局公开，有确定走步的一类棋类进行研制的。而对于多人对弈，随机性的博弈这类问题，至少目前计算机还是难以模拟实现的。

3. 机器翻译所面临的问题

在计算机诞生的初期，就有人提出了用计算机实现自动翻译的设想。由于当时计算机技术水平的限制还无法提供大容量的存储设备和速度足够快的运算部件来处理像文字翻译这样的复杂问题。直到 60 年代机器翻译的研究才进入活跃期。如果对机器翻译不是那么苛刻要求，

而仅仅把它看作计算机的一种辅助翻译,那么现在已经问世的英—俄、日—英、英—汉、日—汉、俄—汉等翻译机,可以应用于某些特定的场所。

机器翻译目前所面临的问题仍然是 1964 年语言学家黑列尔(B·Hillel)所说的构成句子的单词和歧义性问题。

歧义性问题一直是自然语言理解(NLU)中的一大难关。同样一个句子在不同的场合使用,其含义的差异是司空见惯的。因此,要消除歧义性就要对原文的每一个句子及其上下文,寻找导致歧义的词和词组在上下文中的准确意义。然而,计算机却往往孤立地将句子作为理解单位。另外,即使对原文有了一定的理解,理解的意义如何有效地在计算机里表示出来也存在问题。

目前的 NLU 系统几乎不能随着时间的增长而增强理解力,系统的理解大都局限于表层上,没有深层的推敲,没有学习,没有记忆,更没有归纳。

导致这种结果的原因是计算机本身结构的问题和研究方法的问题。现在 NLU 的研究方法很不成熟,大多数研究局限在语言这一单独的领域,而没有对人们是如何理解语言这个问题作深入有效的探讨。

4. 自动定理证明和 GPS 的局限

自动定理证明的代表性工作是 1965 年鲁宾逊提出的归结原理。

归结原理虽然简单易行,但它所采用的方法是演绎,而这种形式上的演绎与人类自然演绎推理方法是截然不同的。基于归结原理演绎推理要求把逻辑公式转化为子句集合,从而丧失了其固有的逻辑蕴涵语义。

前而曾提到过的 GPS 是企图实现一种不依赖于领域知识,求解人工智能问题的通用方法。GPS 想摆脱对问题内部表达形式的依赖,但是问题的内部表达形式的合理性是与领域知识密切相关的。研究结果表明, GPS 可以很好地用于求解某类问题,而对另一些问题,则需要改变问题的内部表达形式。问题内部表达形式的改变,反过来又要求改变 GPS 的内部结构。

不管是用一阶谓词逻辑进行定理证明的归结原理,还是求解人工智能问题的通用方法 GPS,我们都可以从中分析出表达能力的局限性,而这种局限性使得它们缩小了其自身的应用范围。

5. 模式识别的困惑

虽然使用计算机进行模式识别的研究与开发已取得大量成果,有的已成为产品投入实际应用,但是它的理论和方法与人的感官识别机制是全然不同的。人的识别手段——形象思维能力,是任何最先进的计算机识别系统望尘莫及的,正如费根鲍姆教授所说的:“我们认为最能显示人的特点的一切结构精密的符号制造物,如数学和逻辑、拼接基因或在根据仪器推断下的地质实况的能力,计算机处理得最高明,因为知识结构越严密,我们越容易把它编纂起来供计算机使用。另一方面,在现实世界中,生活并不是一项结构严密的任务——一般家畜都能轻而易举地对付,但机器不会,这并不是说它们永远不会,而是说目前不会。”

6. 自动程序设计的困难

自动程序设计仍是自顶向下进行功能分解的一种软件开发方法,从外部功能上进行模拟客观世界,其开发过程是从“做什么”到“如何做”。我们知道,尽管这种程序设计方法并不是什么新的方法,然而,它使程序设计向自动化方向的确迈了一大步。它使程序设计者从烦琐的低级语言程序设计中摆脱出来,使用比较自然的高级语言来完成思维活动。

以往的程序设计活动被认为是一种艺术,在很大程度上体现了设计者的个人技巧。随着软

件工程思想的出现,程序设计成了一种有规可循的思维活动。它的基本思想是将用户表达的非形式说明转化成形式规范,然后再通过编码形成程序,最后再加上有关的说明书,形成实实在在的产品。

按照上述思路进行程序设计,必须具备以下几个条件:

(1)要充分理解用户的意图 这个问题直接关系到所设计的程序能否满足用户的要求。事实上,要想使计算机取代程序设计员是很困难的。因为人们采用的能够真正说明意图的表达形式,计算机很难理解;另一方面计算机容易理解的表达形式,对人类又不适应。

(2)要有恰当的生成程序代码 程序设计的问题除了有可能超出所涉及的领域知识外,程序设计知识也是至关重要的。程序设计既涉及到选择控制结构,又涉及选择数据结构。这样使得程序作有效的选择变得十分困难。即使对用户要求有所理解,要想将这种要求转变成代码也不是那么容易的事情。

(3)要有一个程序设计的具体标准 程序设计的过程是在理解的前提下,以知识为基础的思维活动。单一的过程法、演绎法和变换法都不是程序设计员实际所采用的根本方法,尽管问题分解、逐步求精已被人们广泛使用,然而在实际中却没有一个具体的使用标准。同一个人在不同时间设计的同一个程序,其程序可能不尽相同,这种现象是不足为奇的。这是因为人本身有许多潜在的因素起作用,而绝不是简单映射和变换的关系。

综上所述,想要使计算机进行程序自动设计并非易事。目前在自动程序设计方面采用的方法仍是定理证明法,程序变换法,基于知识的方法,问题求解法和归纳法。对于定理证明法而言,要使用谓词正确表达复杂程序的输入输出,比写程序还难。而且其定义域必须完全公理化,否则证明器就无法产生证明,因而也就无法生成程序。程序变换法仅仅适合于描述清楚易于理解的问题。传统的问题求解可能导致问题的复杂化而难以控制。归纳法很不完全,而基于知识的方法具有与专家系统相同的困难。因此,现在自动程序设计仍然处在初级阶段,理论的不完善和技术的不成熟使之进展十分缓慢。

如果说人工智能诞生后的头十年,主要是一批科学家在实验室中进行人工智能的基本原理和方法的研究,那么从 1965 年费根鲍姆教授领导的研究小组开始研究专家系统 DENDRAL 开始,人工智能从实验室走了出来,进入实际应用时代。

70 年代后期开始,一大批实用型专家系统不断涌现。如用于超大规模集成电路设计的 KBVLSI,自动程序设计系统 PSI;用于遗传学实验设计的 MOLGEN 系统;用于感染病诊断治疗的教学系统(GUZDON)等。这些专家系统的实际运行取得了重大的社会和经济效益,也为人工智能的发展带来了生机。

除此之外,为了加速专家系统的研制速度,人们还先后开发了一批用于建造和维护专家系统的工具系统。例如:Meta-DENDRAL,TEIRESIAS 和 EMYCIN 等。

专家系统的成功所产生的社会影响,不但激起人工智能学术界的研究开发热情,而且也引起了社会各界的兴趣。一个新的研究开发领域——知识工程(Knowledge Engineering)由此应运而生。

1977 年费根鲍姆教授在第五届国际人工智能联合会(IJCAI)会议上提出了知识工程的概念。至此,专家系统已基本成熟。人工智能研究者纷纷围绕开发专家系统进行理论、方法、技术等各方面的研究,一门新兴的学科——知识工程逐渐形成了。

费根鲍姆认为,知识工程是应用人工智能的原理和方法,对那些需要专家知识才能解决的应用难题,提供求解的手段。恰当运用专家知识的获取、表达和推理过程的构成与解释,是设计

基于知识的系统的重要技术问题。这类以知识为基础的系统,就是通过开发智能软件而建立的专家系统。

知识工程这一概念的引入和使用,赋予专家系统研究与开发的活力。

如上所述,专家系统的研究与开发已经取得了重大成果,但是,也应该清楚地认识到专家系统的发展也面临着许多实质性的问题。

费根鲍姆曾说过:专家系统的威力(即才能)来源于知识,但是如果把当代专家系统所以发挥其作用的知识与人类专家解决问题时所依赖的知识作一较深入的分析比较,就会发现至少存在以下几个方面的差异。

(1)人类专家所具备的知识中,存在一些只能意会不可言传的经验性知识,根本不可能借助于符号系统来描述;而纳于专家系统知识库中的知识,却只限于在形式上能够符号化的知识。

(2)人类专家总是通过学习或训练,不断地积累知识,更新知识,即使是在解决问题的实践中,也同时包含了学习/训练过程。人类专家是主动地进行学习,专家系统虽然也可以包含某种学习机制,即与机器学习相结合,但这种学习在本质上是被动的,只能按照规定的模式机械地进行,而缺乏自适应性和自组织能力。

(3)人类专家解决问题的能力不仅来源于领域知识,而且还依赖于背景知识;解决问题所体现的水平,既取决于专业知识的广度和深度,也依赖于背景知识的广度和深度。专家系统的知识库内容则基本上局限于有针对性的领域知识。

(4)人类专家并非仅仅依靠逻辑推理来解决问题。形象思维、联想、直觉、领悟等非逻辑推理的思维活动也在解决问题中起着重要的作用。有时,甚至是决定性的作用,即使是纯抽象思维过程,往往也是演绎和归纳推理交织在一起。专家系统的推理机制往往局限于某种逻辑推理控制策略,可以说完全没有上述非逻辑推理思维活动的成分。

显然,上述这些差异绝不是单纯地依靠计算机技术的进步所能解决的。因为目前人类对于自身具有的智慧的本质还未认识清楚。人们对知识在大脑皮层结构中究竟是怎样表达还缺乏足够的了解。同时,对于错综复杂的大脑神经元组织中,是如何运用知识来进行思维的还有待于科学家们去探索。

二、人工智能研究的进展

人工智能的发展过程表明:人工智能是有史以来最艰难的科学之一。进入 80 年代以来,一方面在知识表示、常识推理、机器学习、分布式人工智能以及智能机器体系结构等基础研究方面取得了令人鼓舞的进展,联结主义在人工神经网络研究方面也取得了可喜的成果;另一方面,人工智能也受到了来自日本第五代计算机研制未能达到预期目标的冲击。这表明传统人工智能所依赖的理论和方法还不能满足人们的期望。这一严酷的事实不得不引起人工智能学术界的注意。于是,人们又重新考虑人工智能研究所遵循的原则,对人工智能的基础进行反思。

1987 年 5 月在麻省理工学院召开的专题讨论会上,人工智能学术界的几位代表人物,阐明了各自对人工智能基础的观点。

1987 年,“Computational Intelligence”杂志发表了 McDermott 的“纯粹理性批判”的论文,次年,又发表了 P·Cheesman 的“计算机理解质疑”的文章。自从 1987 年以来,人工智能的几次学术会议都有关于非单调推理背景的常识表示和推理的辩论。

1991年著名杂志“Artificial Intelligence”(Vol. 47)发表了人工智能基础专辑。在这期专辑中,十多位人工智能代表人物提出了一些思想新颖的论点,并就人工智能有关方面的基础性假设进行了辩论,显然,这场辩论本身就是对人工智能基础理论发展的一种促进。

在实际智能系统的研制过程中,人们经过对传统人工智能的反思,认识到智能系统的研究应该走综合集成的道路。综合集成观点认为智能系统是一个具有社会性的人与机器亲密合作的巨型系统。某些智能系统,如专家系统研究不成功的原因是它所包含的常识知识太少,系统处理知识的能力又太弱。因此智能系统的研制从方法论、基础理论到实现技术均产生了深刻的变化。系统的知识数量再也不是 10^5 数量级,而是以 $10^7\sim10^8$ 为目标;系统的设计方式及系统体系,再也不是“自力”型的孤立系统,而是社会型的分布系统;系统的维护上再不遵守智能系统的一致性假设,而是强调解决矛盾的能力;研制所追求的目标,再不强调系统的自主性,而是强调人的作用。

综合集成智能系统的社会性体现在多学科的交叉。人们认识到人工智能已有的理论基础太薄弱,不能很好地描述人类的智能行为。于是其他学科的学者陆续将本学科发展起来的理论与方法向人工智能渗透。从而导致了目前人工智能研究多学科交叉的形势。例如:

(1)逻辑学中的非单调逻辑日益成为知识表示的基础之一。

(2)吴文俊方法将代数几何的原理引入人工智能,用基于求解非线性代数方程的推理形式丰富了人工智能。

(3)人工神经网络研究的兴起,使得神经生理学和非线性科学渗透到人工智能。

(4)控制论强调机器对实际环境的适应是基于行为的,这种对环境建模的方法是区别于传统的人工智能的。

(5)生物学向人工智能的渗透表现在它强调智能行为可以理解为一种自然选择和进化的过程。

(6)社会学认为智能是一种社会行为。智能表现在社会中各个个体之间的交流之中,并且系统中允许存在有矛盾。随着计算机网络技术的日益成熟,这个思想已成为分布式人工智能研究的理论框架。

(7)系统科学将思维过程看成是一个开放性的系统。在建立这样的系统时,人被看作智能系统的一个有机因素,人与机器在系统科学的指导下构成整个智能系统。

(8)心理学继续向人工智能渗透,一些典型图形的和声音事例对智能影响的研究,将为多媒体技术的实现提供理论依据。

以上各学科对人工智能渗透反映了目前人工智能发展的一种趋势,至于其渗透的结果现在还不明显,还需要时间的考验。

人工智能研究者一直清醒地认识到无论人工智能如何发展,机器终究是不会代替人的。基于这种考虑,人的参与使得智能系统的性能变得十分重要了。另一方面,只有人所承认的机器行为才算是智能行为,近年来,科学家们试图将多媒体技术、虚拟现实(Virtual Reality)技术等用到感知通信的研究中,以满足不同人的需要。这是因为人认识现实世界的方式和途径是多种多样的,它们可以互为补充,以达到较为准确的理解。目前,这方面的研究进展很快,引起人们的高度重视。

习 题 一

1. 什么是人工智能？一个人工智能系统应该具有哪些能力？
2. 什么是智能？它应至少具有哪些知识？
3. 人工智能研究中有哪些学派，它们各自的特点是什么？
4. 人工智能研究的对象和目标是什么？
5. 列举人工智能的应用领域。
6. 叙述图灵试验的过程。
7. 人工智能研究各个发展阶段及其特点是什么？
8. 你认为人工智能作为一门学科，今后的发展方向应该如何？
9. 解释下列名词术语：
 - (1) 知识工程
 - (2) 专家系统
 - (3) 人工神经网络
 - (4) 模式识别
 - (5) 问题求解
 - (6) 自然语言理解
 - (7) 自动定理证明
 - (8) 机器视觉
 - (9) 自动程序设计
 - (10) 机器学习
 - (11) 机器人

第二章 问题求解的基本原理

问题求解是人工智能中的核心概念。所谓问题求解是要求在给定条件下寻求一个能解决某类问题且能在有限步内完成的算法。通常这种方法是通过在某个可能的解答空间寻找一个解来进行的。实际上许多求解方法是采用试探搜索方法实现的。这里包括算法的存在性问题及怎样去控制算法,以降低算法的复杂度问题。人工智能中的问题求解归纳起来主要有两种:(1)状态空间法;(2)问题归纳法。本章将介绍这两种方法。

第一节 状态空间法问题求解

一、问题的状态空间表示

状态空间可用三元组 (S, O, G) 来描述,其中:

S 是状态集合。其中每个元素表示一个状态。状态是某种事实的符号或数据。 S_0 是 S 的非空子集,是问题的初始状态。

G 也是 S 的非空子集,表示目标状态集。它可以是若干具体的状态,也可以是对某些状态性质的描述。

O 是操作算子集,利用它将一个状态转化为另一个状态。

状态空间的一个解是一个有限的操作算子序列,它使初始状态转化为目标状态:

$$S_0 \xrightarrow{O_1} S_1 \xrightarrow{O_2} S_2 \xrightarrow{O_3} \cdots \xrightarrow{O_k} G$$

其中 O_1, \dots, O_k 即为状态空间的一个解,解不一定是唯一的。

状态空间可用有向图表示(如图 2-1),其结点表示状态,结点间的弧表示操作算子。从状态 S_0 使用操作算子 O_1 及 O_2 ,分别使 S_0 转化为 S_1 及 S_2 。如此下去,若 S_4 是 G ,则 O_2, O_4 就是一个解。

用状态空间法求解问题时,首先将所要求的问题表示成状态空间,问题的解就在状态空间中。然后,根据给定的条件,在状态空间中搜索出目标状态,从而求得问题的解。搜索方法的效率将决定问题求解的效率。

例 2-1 梵塔问题

起源于印度传说的梵塔问题是:有三根针和 n 个大小不同的盘片。开始盘片都是叠在第一根针上,从下到上按由大到小的顺序串叠。要求每次只移动最顶上的一个盘片到另一根针上,且大盘不得压在小盘上,直到把所有盘片移到第三根针上。以三圆盘为例,如图 2-2 所示。

用状态 (i, j, k) 表示最大盘片 C 在第 i 根针上,盘片 B 在第 j 根针上,最小盘片 A 在第 k 根

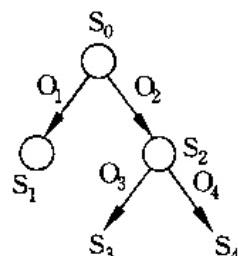


图 2-1 状态空间图

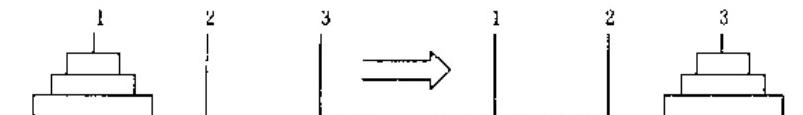


图 2-2 三盘片梵塔

针上。如果同一根针上有二片以上的盘片，则假设较大的在下面。如(1,1,2)表示 C 和 B 在第 1 根针上，且 B 在 C 的上面，而 A 在第 2 根针上。用 $M(N, i, j)$ 表示操作算子，即把盘片 N 从第 i 根针移到第 j 根针上。例如 $M(A, 1, 2)$ 实现的操作是把盘片 A 从第 1 根针移到第 2 根针上，即使状态(1,1,1)变成状态(1,1,2)。而不允许接着进行 $M(B, 1, 2)$ 操作，使状态(1,1,2)变为状态(1,2,2)，因为这违反了小片必须在大片上的规则。

三盘片梵塔的状态空间图表示如图 2-3。

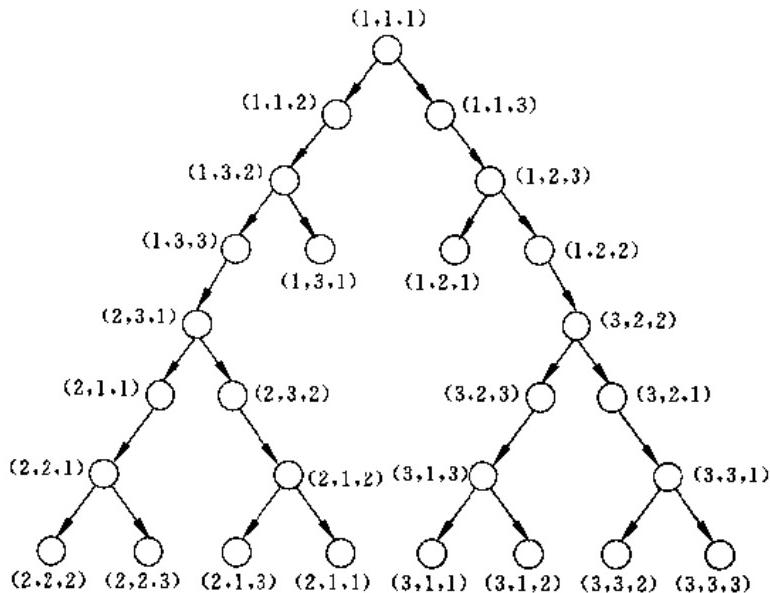


图 2-3 三盘片梵塔的状态空间图表示

二、状态空间的穷搜索法

前已指出，用状态空间来求解问题的过程是构造问题的状态空间，搜索目标状态，直到找到目标状态或确定该问题无解。本节介绍用穷搜索法来求解问题的两种基本方法，即广度优先搜索算法和深度优先搜索算法。

(一) 广度优先搜索算法

所谓广度优先搜索算法是在求解问题的过程中，从初始状态开始按照规则进行第一次各种可能的操作之后，扩展产生了第一级各种新的状态，而后再依次分别地对这些新状态再进行各种可能的操作，又扩展产生各种与日俱增的新状态，如此一直到产生目标状态为止（如果有的话）。其搜索次序可用图 2-4 来表示。

宽度优先搜索算法如下：

数据结构：

Open—是一个先进先出队列,存放称为端结点的待扩展的结点。

Closed—是一个表,存放已被扩展过的结点。

算法步骤:

n 置 0. 判初始结点是否为目标结点,如果是,则算法退出。

这时间题解属特殊情况,即初始结点就是目标结点。否则

1. 把初始结点送 Open 队列。该结点的返回指针值为 n,
2. 如果 Open 队列为空,则没有解,算法失败结束,否则继续下一步。
3. 把 Open 队列中最前面的结点取出来,存入 Closed,并赋结点编号为 n。
4. 把结点 n 加以扩展。如果它没有后续结点,则转 2,否则

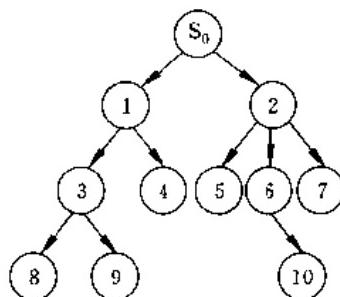


图 2-4 广度优先搜索次序

队首	状态	返回指针
	.	.
	.	.
队尾		

(a) Open队列

编号	状态	返回指针
.	.	.
.	.	.

(b) Closed表

图 2-5 Open 队列与 Closed 表结构

就产生它的所有后续结点,并把它们依次存入 Open 队列的末尾,给出这些结点的返回指针值(父结点编号)为 n。把 n 加 1。

5. 如果刚产生的这些后续结点中存在一个目标结点,则找到一个解。解可从目标结点开始直到初始结点的返回指针中得到,算法成功退出。否则转 2 继续。

在算法中,当新产生的结点状态与某个已产生结点状态相同的话,则广度优先搜索算法能在状态空间图中找到一条从起始结点到目标结点的最佳路径,即长度最短的路径。用广度优先搜索算法求解三盘片梵塔问题,通过不断扩展可得图 2-3 所示的状态空间图,得到的解为图中最右边从起始结点到目标结点的路径。如果解不存在,则该算法就产生一个有限图,算法宣告失败而退出,或者将永无止境地产生一个无限图。

(二)深度优先搜索算法

深度优先搜索算法是把最近刚产生的结点优先扩展,直到达到一定的深度限制。若未找到目标或无法再扩展时,再回溯到另一结点继续扩展。如图 2-6 所示,图中先沿结点 1,2,3 进行扩展,在结点 3 无后继结点可产生,就回溯到结点 2 沿着 4 继续扩展,并产生新的结点 5。若 5 还不是目标结点,则回溯到结点 1 沿着 6 继续扩展。从中可体会方法的名称中“深度”的含义。

为描述图中结点的深度,给出树的结点的深度定义如下:

(1) 根结点的深度为 0

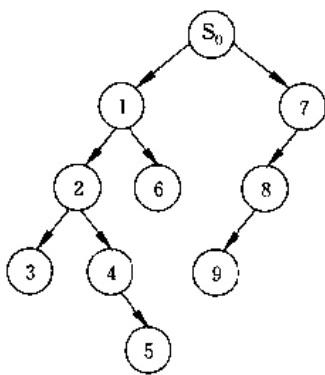


图 2-6 深度优先搜索次序

(2)除根结点外,其他结点的深度是其父结点的深度加 1。

在这种搜索方法中,可能沿着一条路径经过许多次扩展,得不到解,或得到的不是最优解。为加快寻得解的过程,提高算法效率,可给出一个深度界限,规定扩展的最深结点不能超过这个界限。当某路扩展已达到深度界限时,应返回扩展。

深度优先算法如下:

数据结构:

Open 是一个先进后出的堆栈,存放待扩展结点。

Close——是一个表,存放已被扩展过的结点。

D——深度界限值。

算法步骤:

(1) n 置 0,判断初始结点是否为目标结点,如果是,则算法退出。这时问题解属特殊情况,即初始结点就是目标结点。否则把初始结点送 Open。该结点的返回指针值为 N 。

(2)如果 Open 为空,则没有解,算法失败结束,否则继续下一步。

(3)把 Open 中的栈顶结点取出,存入 Close 表,并赋结点编号为 N 。

(4)如果结点 n 的深度等于深度限度的话,就转到 2,否则继续下一步。

(5)把结点 n 加以扩展。如果它没有后继结点,则转 2,否则就产生它的所有后继结点,计算后继结点的深度,并把它们依次压入 Open 堆栈,给出这些结点的返回指针(父结点编号)为 n 。并把 n 加 1。

(6)如果刚产生的这些后继结点中有一个是目标结点,则找到一个解。解可从目标结点开始直到初始结点的返回指针中得到,算法成功退出。否则转 2 继续。

在深度优先搜索算法中,深度限度取得过小,就会找不到目标结点而失败。一旦失败,还可增大深度限度,重新搜索。如果深度限度取得过大,则会既费时又费空间。因此,深度限度要取得适当。算法中解的路径长度总是小于或等于深度限度。

三、启发式搜索法

穷搜索方法只适用不太复杂的任务。对 NP 完全类问题,用穷搜索方法在搜索过程中将扩展太多的结点,类似传统的算法那样,难以避免组合爆炸的厄运。

启发式搜索法的基本思想是在搜索路径的控制信息中增加关于被解问题的某些特征,用于指导搜索向最有希望到达目标结点的方向前进。它与穷搜索不同,它一般只要知道问题的部分状态空间就可以求解该问题,搜索效率较高。

(一)评估函数和启发信息

与被解问题的某些特征有关的控制信息(如解的出现规律、解的结构特征等)称为搜索的启发信息。它反映在评估函数中。评估函数的作用是估计待扩展各结点在问题求解中的价值。一般而言,估价一个结点的价值,应考虑两个因素:为得到解,已经付出的代价和将要付出的代价。评估函数 $f(n)$ 可定义为:

$$f(n) = g(n) + h(n)$$

这里 $g(n)$ 表示迄今为止搜索已产生的从初始结点 S_0 到 N 的实际代价, $h(n)$ 表示从 N 到目标

结点 S 的估计代价, $h(n)$ 称为启发函数, 它体现了搜索的启发信息。 $g(n)$ 可根据已生成的搜索树实际计算出来, 而 $h(n)$ 需要根据具体问题进行估计。有了 $f(n)$ 就可以按 $f(n)$ 的大小来安排待扩展结点的次序, 选择 $f(n)$ 最小的结点先进行扩展。

(二)一般图启发式搜索算法 A

- (1) 建立一个只含初始结点 S_0 的搜索图 G , 把 S_0 放入 Open 表, 并计算 $f(S_0)$ 值
- (2) 如果 Open 表是空的, 则失败; 否则, 继续下一步。
- (3) 从 Open 表中取出 f 值为最小的结点(第一个结点), 置于 Close 表中, 给这个结点编号为 n 。
- (4) 如果 n 是目标结点, 则得解, 算法成功退出。此解可从目标结点开始直到初始结点的返回指针中得到。否则, 继续下一步。
- (5) 扩展结点 n 。如果它没有后继结点, 则立即转(2); 否则, 生成 n 的所有后继结点集 $M = \{m_i\}$, 把 m_i 作为 n 的后继结点添入 G 。并计算 $f(m_i)$ 。
- (6) 若 m_i 未曾在 G 中出现过, 即未曾在 Open 或 Closed 表中出现过, 就将它配上刚计算过的 $f(m_i)$ 值和返回到 n 的指针, 并把它们放入 Open 表中。
- (7) 若 m_i 已在 Open 表中, 则需要把其原来的 g 值与现在刚计算过的 g 值相比较: 若前者不大于后者, 则不做任何更改; 若前者大于后者, 则将 Open 表中该结点的 f 值更改为刚计算的 f 值, 返回指针更改为 n 。
- (8) 若 m_i 已在 Closed 表中, 但 $g(m_i)$ 小于原先的 g 值, 则将表中该结点的 g , f 值及返回指针进行类似第(7)步的修改。并要考虑修改表中通向该结点的后裔结点的 g , f 值及返回指针。
- (9) 按 f 值自小至大的次序, 对 Open 表中的结点重新排序。
- (10) 返回第(2)步。

上述算法 A 生成一个明确的图 G (称为搜索图)和一个 G 的子集 T (称为搜索树), 图 G 中的每个结点也在树 T 上。搜索树是由第 7 步设置的返回指针来确定的。 G 中的每一个结点(除 S_0 外)都有一个只指向 G 中一个父辈结点的指针。该父辈结点就是树中那个结点的唯一的父辈结点。对任何结点的单独一条明显路径是用搜索树 T 来定义的。

算法中(7),(8)步保证对每一个扩展的新结点, 其返回指针的指向是已产生的路径中代价最小的。下面举例来说明:

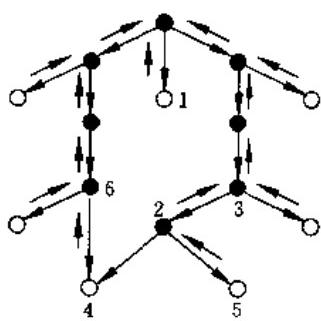


图 2-7(a) 结点 1 扩展之前的
搜索图和搜索树

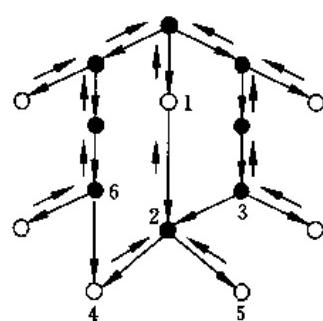


图 2-7(b) 结点 1 扩展之后的
搜索图和搜索树

设搜索算法已生成如图 2-7(a)所示的搜索图和搜索树(带返回指针的部分)。图中实心圆点表示在 closed 表中的结点, 空心圆点表示在 open 表中的保护层的结点。图 2-7(a)是扩展结

点 1 之前的搜索图的搜索树;而图 2-7(b)是扩展结点 1 之后的搜索图和搜索树。

当结点 1 扩展时,产生结点 2。而结点 2 是 closed 表中的结点,但结点 2 的新 $g(2)$ 值为 2 (设相邻两结点之间的代价均为 1),而原来 $g(2)$ 值为 4,故将结点 2 指向结点 1,将原来指向结点 3 的指针断开。同时要考虑修改 2 的后继结点 4 的返回指针和 $g(4)$ 值,将结点 4 指向结点 2,并把原来指向 b 的指针断开。

现在用算法 A 来解重排九宫问题。

在 3×3 的井字九宫格棋盘上摆有 8 个将牌,分别标有 1~8 的一个数码。棋盘上尚有一个空格,允许其周围的将牌向空格移动。这样通过移动将牌可以不断变换将牌的布局。现给定如图 2-8 所示的两种布局,一种为初始状态,另一种为目标状态。问如何移动将牌,初始状态将变成目标状态?

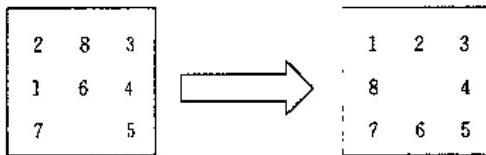


图 2-8 九宫重排问题的举例

设该问题的评估函数为:

$$f(n) = g(n) + w(n)$$

其中, $g(n)$ 是搜索树中从起始结点到 n 的路径长度,而 $w(n)$ 代表结点 n 的格局与目标结点的格局相比,位置不符的将牌数目。例如在图 2-9 中, $w(S_0)=5$, $w(2)=3$, $w(3)=4$, $w(S_g)=0$ 等等。对于起始结点的评估函数为 $f(S_0)=g(S_0)+w(S_0)=0+5=5$ 。

应用算法 A 的结果如图 2-9 所示。图中各结点的 f 值,用数字加圆圈表示各结点的 w 值,不加圆圈的数字表示结点扩展的顺序。

(三) 启发式搜索算法 A*

算法 A 不能保证当图中存在从起始结点到目标结点的最短路径时,一定能找到它。那么如何才能确保找到这条最短路径呢?为此,需要定义评估函数 f^* :

$$f^*(n) = g^*(n) + h^*(n)$$

其中 $g^*(n)$ 为起始结点到结点 n 的最短路径值, $h^*(n)$ 是 n 到目标结点的最短路径。这样, $f^*(n)$ 就是从起点出发通过 n 到达目标状态的最佳路径的总代价估值。

算法 A 中的 $g(n)$ 和 $h(n)$ 是 $g^*(n)$ 和 $h^*(n)$ 的近似估价。 $g(n) \geq g^*(n)$,仅当搜索已发现到达 n 的最佳路径时它们才相等。如果对所有结点 n 都有满足 $h(n) \leq h^*(n)$,则称此时的算法 A 为算法 A^* 。算法 A^* 保证只要最短路径存在,就一定能找出这条路径。由于算法 A^* 对任何存在最短路径的图,总能找到最短路径,所以称算法 A^* 是可接纳的。

显然,对上节的九宫重排问题,由于 $W(n) \leq h^*(n)$,故例中找出的是最短路径。

应当指出,同一问题启发函数 $h(n)$ 可有多种设计方法。在九宫重排问题中,还可以使 $h(n) = p(n)$, $p(n)$ 表示结点 n 的每一将牌与其目标位置之间的距离总和。显然, $w(n) \leq p(n) \leq h^*(n)$,相应的搜索过程也是 A^* 算法。然而, $p(n)$ 比 $w(n)$ 有更强的启发信息,因而由 $h(n) = p(n)$ 构造的启发式搜索树,它所含结点数比 $h(n) = w(n)$ 构造的搜索树结点数要少。

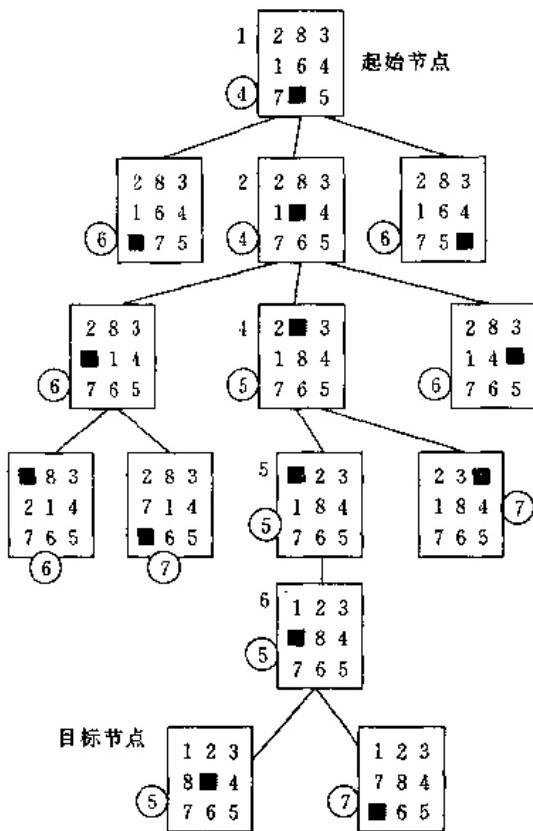


图 2-9 重排九宫问题的算法 A 解

第二节 问题归约法

问题归约法是不同于状态空间法的另一种问题描述和求解的方法。

一、问题归约描述

问题归约也可用一个三元组 (S_0, O, P) 来描述, 其中:

S_0 是初始问题, 即要求解的问题。

P 是本原问题集, 其中的每一个问题是不证明的, 自然成立的, 如公理、已知事实等, 或已证明过的。

O 是操作算子集, 通过一个操作算子把一个问题化成若干个子问题。

这种表示方法是由问题出发, 运用操作算子产生一些子问题, 对子问题再运用操作算子产生于问题的子问题, 这样一直进行到产生的问题均为本原问题, 则问题得解。

所有问题归约的最终目的是产生本原问题。问题归约法是比状态空间法更适用的问题求解方法, 如果在归约法中, 每运用一次操作算子, 只产生一个子问题, 则就是状态空间法。

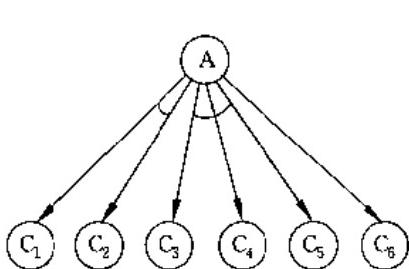


图 2-10 子问题替换集合的结构

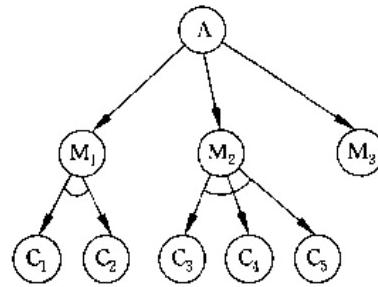


图 2-11 各结点后继只含一个 K 连接弧的与或图

二、与或图表示

用与或图可以方便地把问题归约为子问题替换集合。例如，假设问题 A 既可通过问题 C_1 与 C_2 ，也可通过问题 C_3 、 C_4 和 C_5 ，或者由单独求解问题 C_6 来解决，如图 2-10 所示。图中各节点表示要求解的问题或子问题。

问题 C_1 和 C_2 构成后继问题的一个集合，问题 C_3 、 C_4 和 C_5 构成另一个后继问题集合；而问题 C_6 则为第三个集合。对应于某个给定集合的各结点，用一个连接它们的圆弧来标记。图 2-10 中连接 C_1 、 C_2 和 C_3 、 C_4 、 C_5 的圆弧叫 2 连接弧和 3 连接弧。一般而言，这种弧叫 K 连接弧，表示对问题 A 作用某个操作算子后产生 K 个子问题。

由结点及 K 连接弧组成的图，称为与或图，当所有 K 均为 1 时，就变为普通图。

可以引进某些附加结点，以便使含有一个以上后继问题的每个集合能够聚集在它们各自的父辈结点之下。这样图 2-10 就变为图 2-11 所示的结构了，每个结点的后继只包含一个 K 连接弧。弧连接的子结点叫与结点，如 C_1 、 C_2 及 C_3 、 C_4 、 C_5 。 $K=1$ 的连接弧连接的子结点叫做或接点。

当用与或图表示问题归约方法时，其始结点对应于初始问题，第一个 K 连接弧对应于使用了某个操作算子，相应的子结点就表示用该算子后产生的子问题，与或图中的叶结点表示本原问题。

在下面的讨论中，假设与或图中每个结点，只包含一个 K 连接弧连接的子结点。

在与或图上执行的搜索过程，其目的在于表明起始结点是有解的。与或图中一个可解结点可递归地定义如下：

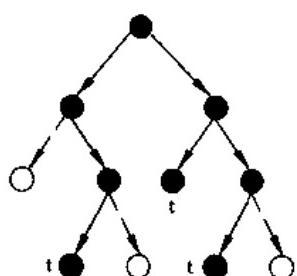


图 2-12 与或图及其解图

(1) 叶结点是可解结点；
(2) 如果某结点有或子结点，那么该结点可解当且仅当至少有一个结点为可解结点。

(3) 如果某结点有与子结点，那么该结点可解当且仅当所有子结点均为可解结点；

不可解结点可递归定义如下：

(1) 没有后裔结点的非终止结点是不可解结点；
(2) 或结点是不可解结点，当且仅当它的所有子结点都是不可解结点；

(3) 与结点是不可解结点，当且仅当它的子结点中至少有一个是不可解结点。

图 2-12 的例子中,可解结点用黑圆点表示,不可解结点用圆圈表示。能导致初始结点可解的那些可解结点及有关连线组成的子图称该与或图的解图。

三、AO* 算法

为了在与或图中找到解,需要一个类似于 A^* 的算法,Nilsson 把它称为 AO^* 算法。 AO^* 算法和 A^* 算法是不同的,其主要区别在于:

1. AO^* 算法要能处理与图,它应找出一条路径,即从该图的开始结点出发到达代表解状态的一组结点。注意,可能需要到达多个解状态,因为一与弧的每条臂要引至它自身的结点。

为弄清为什么 A^* 算法不足以搜索与或图,可以考察图 2-13(a)。扩展顶点 A 产生两个子结点集合,一个为结点 B ,另一个由结点 C, D 组成。在每个结点旁边的数表示该结点 f 值。为简单起见,假定每一操作的耗费一致。设带一个后继结点的耗费为 1。若查看结点并从中挑选一个带最低 f 值的结点扩展,则要挑选 C 。但根据现有信息,最好开发穿过 B 的那条路径,因 C 得用 D ,其总耗费为 9,即($D+C+2$);而穿过 B 的耗费为 6。问题在于下一步要扩展结点的选择不仅依赖于那一结点的 f 值,而且取决于那一结点是否属于从初始结点出发的当前最短路径的一部分。对此,图 2-13(b)更加清楚。按 A^* 算法,最有希望的结点是 G ,其 f 值为 3。 G 结点是 C 的后继。 C 也是 B, C, D 中最有希望的结点,其总耗费为 9。但 C 不是当前最短路径的一部分,因用 C 得用 D ,而 D 的耗费为 27。因此不应扩展 G ,而应考察 E 和 F 。

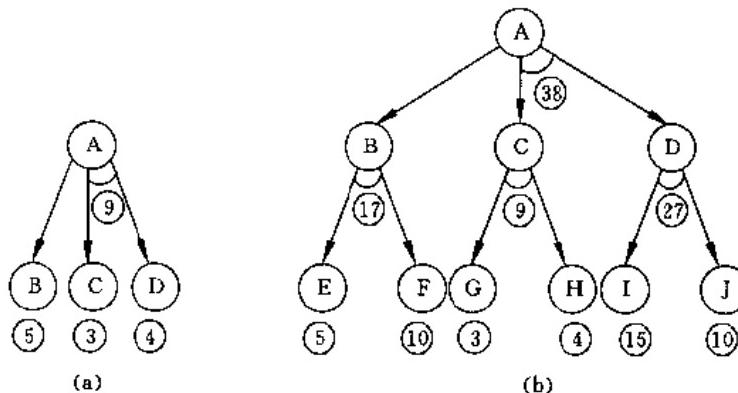


图 2-13 与或图

由此可见,在扩展搜索一与或图时,每步需做三件事:

- (1) 遍历图,从初始结点开始,顺沿当前最短路径,积累在此路径上但未扩展的结点集。
- (2) 从这些未扩展结点中选择一个并扩展之。将其后继结点加入图中,计算每一后继结点的 f 值(只需计算 h ,不管 g)。
- (3) 改变最新扩展结点的 f 估值,以反映由其后继结点提供的新信息。将这种改变往后回传至整个图。在往后回攀图时,每到一结点就判断其后继路径中哪一条最有希望,并将它标记为目前最短路径的一部分。这样可能引起目前最短路径的变动。这种往回攀图传播修正耗费估计的工作在 A^* 算法中是不必要的,因为只需考察未扩展结点。但现在必须考察已扩展结点以便挑选目前最短路径。于是,其值是目前最佳估计这一点很重要。

下面列举图 2-14 中的例子来说明此过程。在第一步, A 是唯一结点,因此它在目前最短路径的末端。扩展 A 后得结点 B, C 和 D ,因为 B 和 C 的耗费为 9,到 D 的耗费为 6,所以把到 D

的路径标志为出自 A 的最有希望的路径(被标志的路径在图中用箭头指出)。

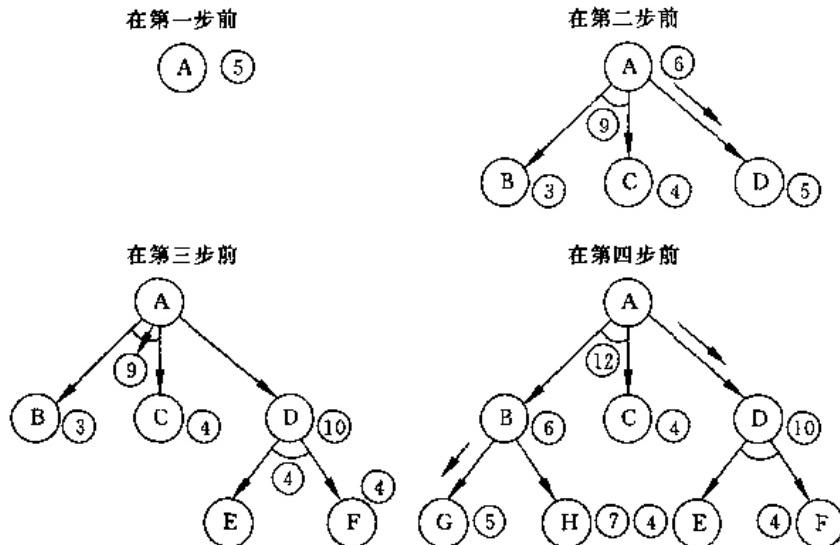


图 2-14 AO^* 算法的运行

在第二步选择 D 扩展。从而得到一新路径, 即得到 E 和 F 的与结点集, 其复合耗费估计为 10, 故将 D 的 f 值修改为 10。往回退一层发现, A 到 B, C 与结点集的耗费为 9, 所以, 从 A 到 B, C 是当前最有希望的路径。第三步扩展 B 结点, 得结点 G, H , 且它们的耗费分别为 5、7。往回传其 f 值后, B 的 f 值改为 6(因为 G 的弧最佳)。往回攀一层后, A 到 B, C 与结点集的耗费改为 12, 即 $(6+4+2)$ 。此后, D 的路径再次成为更好的路径, 所以将它作为目前最短路径, 并在第四步扩展 E 和 F , 继续该过程直至找到一解, 或所有路径都走入死胡同, 从而指出无解存在为止。

2. 如果有些路径通往的结点是其他路径上的与结点扩展出来的结点, 那么不能独立于这些路径去考虑某些从结点到结点的个别路径。 A^* 算法中, 从一结点到另一结点的预期路径总是带最低耗费的路径, 但在 AO^* 算法中, 情况并非如此。

考虑图 2-15(a)中的例子。图中结点已按生成它们的顺序给了序号。现假定下步要扩展结点 10, 其后继结点之一为结点 5, 扩展后的结果如图 2-15(b)所示。到结点 5 的新路径比通过 3 到 5 的先前路径长。但因为要穿过 3 的路径通向解, 还得要穿过 4 的路径通向解, 而已知不存在穿过 4 的解, 所以穿过 10 的路径好一些。

3. AO^* 仅对保证不含任何回路的图操作。作这种保证是因为存储一条回路路径绝无必要。这样的路径代表了一条循环推理链。例如, 在证明数学定理时会出现这样的问题, 可显示: 能证 Y 就能证 X ; 再显示: 能证 X 就能证 Y 。但这样的循环路径不可能构造出证明。所以, 从图中省掉回路路径不会冒漏掉解的危险。虽然在搜索图中没有回路能简化搜索算法的某些部分, 但它又把另一部分弄复杂了。当生成一结点并发现它已在图中时, 就得检查已在图中的那个结点是不是正扩展结点的祖先。仅当不是祖先时(于是回路不会建立), 才把最近发现的到此结点的路径加到图中。此检查能避免生成图 2-15(c)那样的图。

现在可对 AO^* 算法进行描述。

在 A^* 算法中用了两张表: OPEN 表和 CLOSED 表。 AO^* 算法只用一个结构 G , 它表达了至今已明显生成的部分搜索图。图中每一结点向下指向其直接后继结点, 向上指向其直接前趋结点。同图中每一结点有关的还有 h 值, 它估计了从该结点至一组解结点那条路径上的耗费。

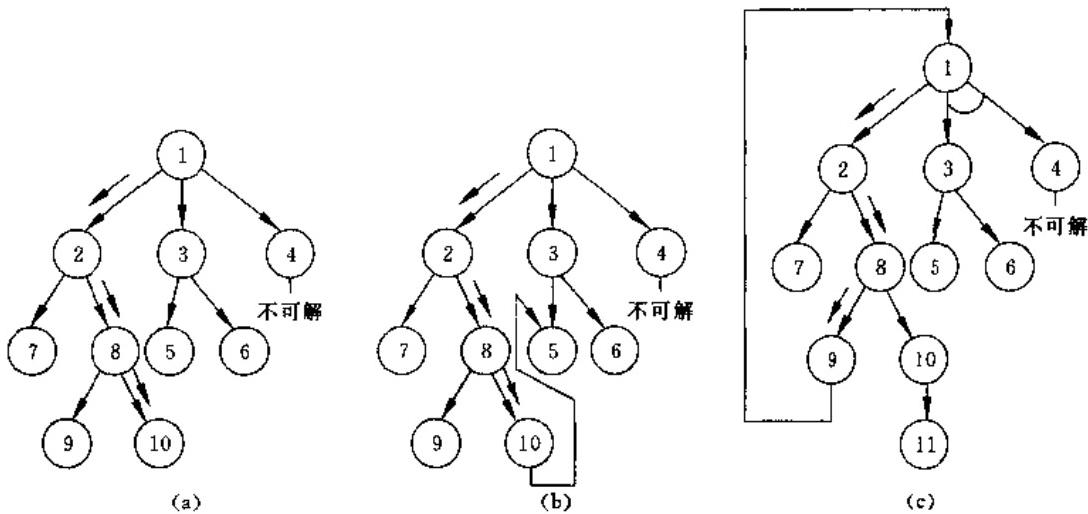


图 2-15 长路径可能更好

与 A^* 算法不同, 这里不存在 g (从开始结点到当前结点的耗费), 不可能计算这个值, 因为有多条路径到达同一状态。而且这个值的计算也没有必要, 因总是自顶向下遍历已知最好的路径, 从而保证只有在最短路径上的结点可能被扩展。所以只用 h 来估计一结点到解的耗费。

AO^* 算法还需用一个称为 FUTILITY 的值。若一解的估计耗费变得大于 FUTILITY 的解, 则要放弃该搜索。FUTILITY 应选得相当于一个阈值, 使得耗费 FUTILITY 之上得任一解即使能找到也昂贵的无法应用。

下面给出 AO^* 算法

- (1) 设 G 仅由代表开始状态的结点组成(称此结点为 INIT), 计算 $h(\text{INIT})$ 。
- (2) 在 INIT 标为 SOLVED(成功)之前, 或 INIT 的 h 值变得大于 FUTILITY(失败)之前, 重复下述过程:
 - a. 跟踪始于 INIT 的已带标记的弧, 挑选出现在此路径上但未扩展的结点之一扩展, 称新挑选的结点为 NODE。
 - b. 生成 NODE 的后继结点, 则对不是 NODE 祖先的每一后继结点(称 SUCCESSOR)应做下述事项:
 - (i) 把 SUCCESSOR 加到图 G 中。
 - (ii) 如果 SUCCESSOR 是一目标结点, 那么将其标记为 SOLVED, 并赋 0 作为 SUCCESSOR 的 h 值。
 - (iii) 若 SUCCESSOR 不是目标结点, 则计算它的 h 值。
 - c. 将最新发现的信息向图的上部回传, 具体做法是: 设 S 为一结点集, 它含有已作了 SOLVED 标记的结点, 或包含其 h 值已经改变因而需要回传至其父结点的那些结点。置 S 的初值为 NODE。重复下述过程, 直至 S 为零:
 - (i) 从 S 中挑选一个结点, 该结点在 G 中的子孙均不在 S 中出现(换句话说, 保证对于每一正在处理的结点, 是在处理其任一祖先之前来处理该结点的), 称此结点为 CURRENT 并把它从 S 中去掉。
 - (ii) 计算始于 CURRENT 的每条弧的耗费。每条弧的耗费等于在该弧末端每一结点的 h 值之和加上该弧本身的耗费。从刚刚计算过的始于 CURRENT 的所有弧的耗费中选出极小耗

费作为 CURRENT 的新 h 值。

(iii) 把在上一步计算出来的带极小耗费的弧标记出始于 CURRENT 的最佳路径。

(iv) 如果穿过新的带标记弧与 CURRENT 连接的所有结点均标为 SOLVED，则把 CURRENT 标为 SOLVED。

(v) 若 CURRENT 已标为 SOLVED，或 CURRENT 的耗费刚才已经改变，那么应把其新状态往回传至图。因此，要把 CURRENT 的所有祖先加到 S 中。

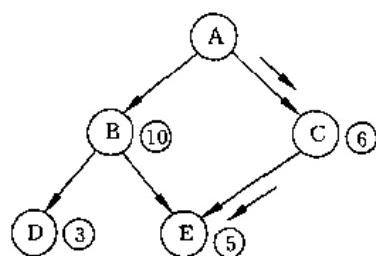


图 2-16 一次无用的逆向传播

关于该算法有几点值得注意。在(2)c. (v)步中，将其耗费已经改变的一结点的祖先，都加到其需修正耗费的结点集，即本算法将把该结点的所有祖先插进集合中，这可能导致穿过大量不很好的路径，往上传递这种耗费改变。例如，图 2-16 中穿过 C 的路径总优于穿过 B 的路径，因而在穿过 B 的路径上所作的工作就是一种浪费。倘若修正了 E 的耗费，并且此改变不穿过 B 向上传递，而只穿过 C 向上传递，则 B 可能会造成 B 看起来更好些的假象。例如，如果作为

扩展结点 E 的结果，将 E 的耗费修改为 10，那么 C 的耗费将修改为 11。如果仅做到这一点，那么当考察 A 时，穿过 B 的路径仅耗费 11，而穿过 C 的路径却耗费 12，从而错误地将穿过 B 的路径作为最有希望的路径。对本例而言，这种错误可在下次扩展 D 时检测出来，若 D 的耗费变了并回传至 B ，则要重新计算 B 的耗费。然后，将 B 的新耗费回传至 A ，从而使穿过 C 的路径再次更好。以上过程只是浪费了扩展 D 的时间。但是如果已经改变耗费的结点处于搜索图中很低的层次，则这种错误可能永远也发现不了，图 2-17(a)就是一例。若照图 2-17(b)所示修正 G 的耗费，并且不立即将此改变回传至 E ，则不会记录此改变，并且可能发现穿过 B 的一个非最佳解。

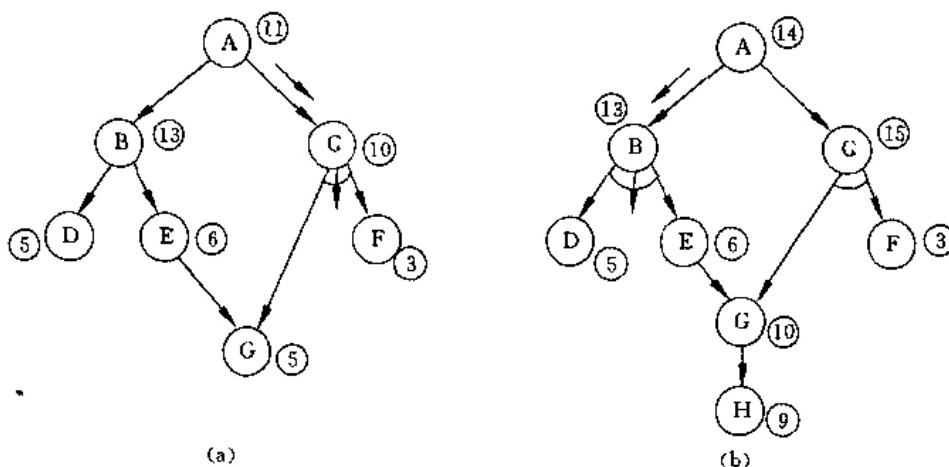


图 2-17 一次必要的逆向传播

此外， AO^* 算法没有考虑子目标间的任一交互。图 2-18 就是一例简单的失策。

假定结点 C 和结点 E 最终通向一个解， AO^* 算法将报告包括两者的一个完全解。与或图指出，为求解 A ，需要求 C 和 D ，但此算法把 D 的解与 C 的解视为完全不同的过程。只要看一下 D 的后继结点就会知道 E 是最佳路径。但毫无疑问 C 是必须的。因此，最好也用它满足 D ，

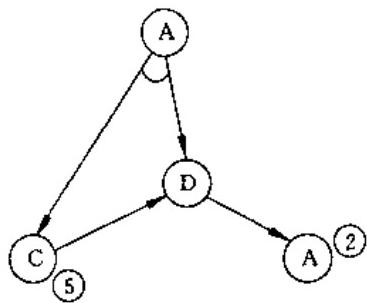


图 2-18 交互的子目标

第三节 博弈树搜索

这里讲的博弈是二人博弈，即由两人对垒，轮流依次走步，每一方都知道对方已经走过的棋步和以后可能走的棋步，双方最终将有胜负之分或者为和局，这类博弈如一字棋、象棋、围棋等。

先来看一个例子，假设有七个钱币，任一选手只能将已分好的一堆钱币分成两堆个数不等的钱币，两位选手轮流进行，直到每一堆都只有一个或两个钱币，不再能分为止，哪个遇到不能再分的情况，则就为输。

用数字序列加上一个说明表示一个状态，其中数字表示不同堆中钱币的个数，说明表示下一步由谁来分，如(7,MIN)表示只有一个由七个钱币组成的堆，由 MIN 走，MIN 有三种可供选择的分法，即(6,1,MAX),(5,2,MAX),(4,3,MAX)其中 MAX 表示另一对手走，不论哪一种方法，MAX 在它基础上再作符合要求的再分，整个过程如下图 2-19 所示。在图中已将双方可能的分法完全表示出来了，而且从中可以看出，无论 MIN 开始时怎么走法，MAX 总可以获胜，取胜的策略用双箭头表示。

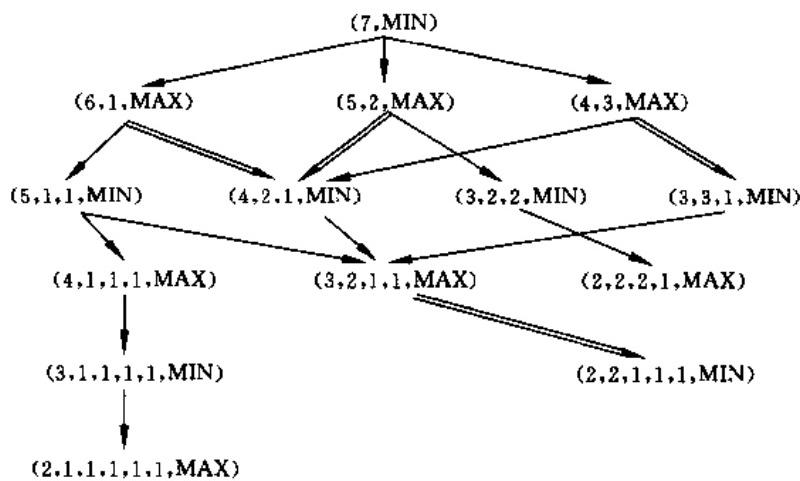


图 2-19 分钱币的博弈

实际的情况没有这么简单，任何一种棋都不可能将所有情况列尽，因此，只能模拟人“向前看几步”，然后作出决策，决定自己走哪一步最有利，也就是说，只能给出几层走法，然后按照一定的估算方法，决定走一步棋。

一、极大极小过程

下面假定由 MAX 选择走一步棋,MAX 如何来选择一步好棋呢? 极大极小过程就是一种方法,这种方法的思想是,对格局给出一个估价函数,因此每一具体格局由估价函数可得一值,值越大对 MAX 越有利,反之,越不利。开始,对给定的格局 MAX 给出可能的走法,然后 MIN 对 MAX 的每一走法,又给出可能的走法,这样进行若干次,得到一组端结点,端结点对应的格局是由 MIN 走后得到的,对每个端结点算出它的估价函数值,然后由底向上逐级计算倒推值,在 MIN 处取估值最小的格局,在 MAX 处取估值最大的格局。一直到 MAX 的开始格局,取估值最大的格局作为 MAX 要走的一步。

图 2-20 表示了向前看两步,共四层的博弈树,用 \square 表示 MAX,用 \circ 表示 MIN,端结点上的数字表示它对应的估价函数的值。在 MIN 处用圆弧连接,0 用以表示其子结点取估值最小的格局。

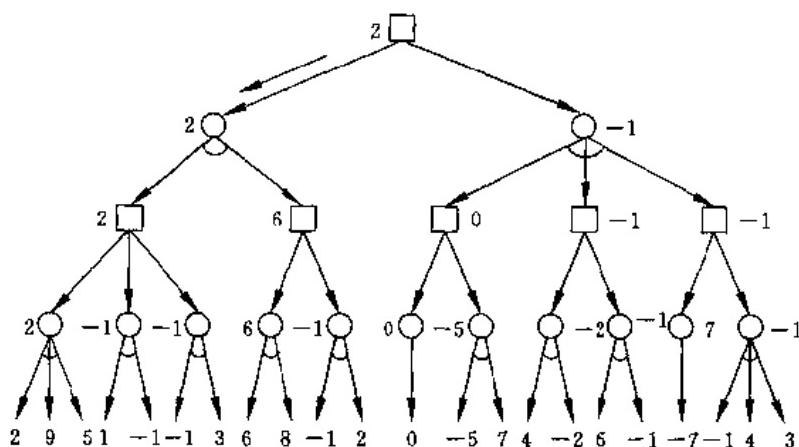


图 2-20 四层博弈树

图中结点处的数字,在端结点是估价函数的值,称它为静态值,在 MIN 处取最小值,在 MAX 处取最大值,最后 MAX 选择箭头方向的走步。

利用一字棋来具体说明一下极大极小过程,不失一般,设只进行两层,即每方只走一步(实际上,多看一步将增加大量的计算和存储)。

估价函数 $e(p)$ 规定如下:

1. 若格局 p 对任何一方都不是获胜的,则

$e(p) = (\text{所有空格都放上 MAX 的棋子之后,MAX 的三个棋子所组成的行、列及对角线的总数}) - (\text{所有空格都放上 MIN 的棋子后,MIN 三个棋子所组成的行、列及对角线的总数})$

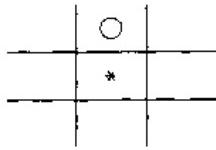
2. 若 p 是 MAX 获胜,则

$$e(p) = +\infty$$

3. 若 p 是 MIN 获胜,则

$$e(p) = -\infty$$

因此,若 p 为



就有 $e(p) = 6 - 4 = 2$, 其中 * 表示 MAX 方。

在生成后继结点时, 可以利用棋盘的对称性, 所以下列格局看成是相同的格局。

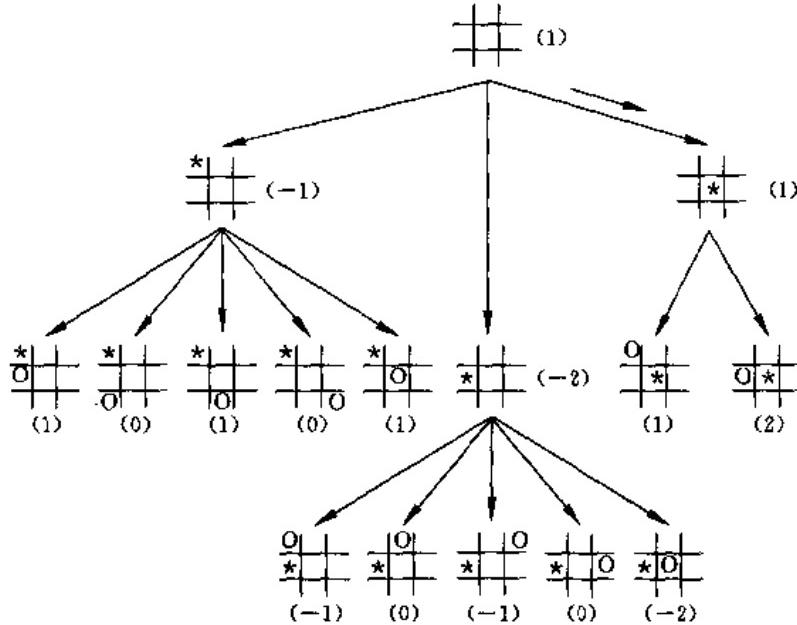
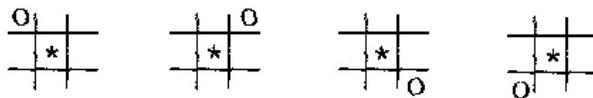
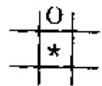


图 2-21 一字棋博弈的极大极小过程

图 2-21 给出了 MAX 最初一步走法的搜索树, 由于 * 放在中间位置有最大的倒推值, 故 MAX 第一步就选择它。



MAX 走了箭头指向的一步, 假如 MIN 将棋子走在 * 的上方, 得到



下面 MAX 就从这个格局出发选择一步, 做法与图 2-21 类似, 直到某方取胜为止。

二、 $\alpha-\beta$ 过程

刚才讨论的极大极小过程先生成一棵博弈搜索树, 然后再进行估值的倒推计算, 将两个过程完全分离, 这种分离是低效率的。重要原因是是否可以在博弈树生成的同时完成端结点的估值及倒推计算, 以减少搜索的次数, 这就是下面要讨论的 $\alpha-\beta$ 过程。

现将图 2-21 的左边一部分重画在图 2-22 中。

前面实际上类似于宽度优先搜索, 将每层格局均生成, 现在用深度优先搜索来处理, 比如结点 A 处, 若已生成五个字结点, 并且 A 处的倒推值等于 -1, 我们将此下界叫做 MAX 结点的 α 值, 即 $\alpha \geq -1$, 现在轮到结点 B, 产生它的第一个后继结点 C, C 的静态值为 -1, 可知 B 处的倒推值 $\beta \leq -1$, 此上界 MIN 结点的 β 值, 即 B 处 $\beta \leq -1$, 这样, 虽然 B 点最终的倒推值可能小于 -1, 但绝不可能大于 -1, 因此, B 结点的其他后继结点的静态值不必计算, 自然不必再生成, 反正 B 决不会比 A 好, 所以通过倒推值的比较, 就可以减少搜索的工作量, 在图 2-22 中即作为 MIN 结点 B 的 β 值小于等于 B 的先辈 MAX 结点 S 的 α 值, 则 B 的其他后继结点就可以不必再生成。

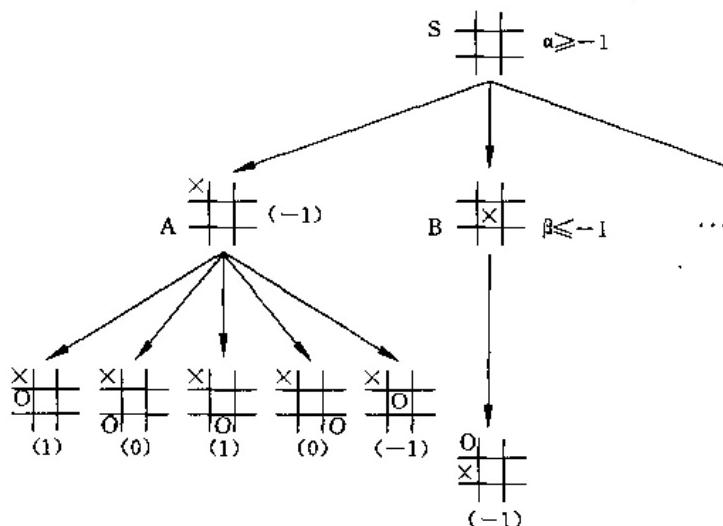


图 2-22 一字棋博弈的 $\alpha-\beta$ 过程

图 2-22 表示了 β 值小于等于父结点的 α 值的情况, 实际上当某个 MIN 结点的 β 值不大于它先辈的 MAX 结点的值, 则 MIN 结点就可以终止向下搜索, 如图 2-23。

若结点 A 搜索了它左端后得 $\alpha \geq 0$, 然后沿着 B、C 搜索到 E 时得到 C 处 $\beta \leq -1$, 此时 C 的 β 小于等于 A 处的 α 值, C 点处的倒推值不可能传到 A, 结点 C 处的其他后继结点可不必产生。

图 2-24 则显示了另一种情况, 即在 MAX 结点 C 处, 搜索完 E、F 后得到 $\alpha \geq 2$, 又设从 B 左边得到 B 处 $\beta \leq 0$, 则 C 处的 α 值大于等于它的先辈 MIN 结点 B 的 β 值, C 以下的结点 D、G、H 可不产生。原因同样是因为此时 C 的倒推值已肯定不会影响 B 处的最后倒推值。

总之, MAX 结点的 α 值不下降, 而 MIN 结点的 β 值不上升, 所以可进行上述剪枝, 一般讲我们可把中止搜索, 即剪枝的规则表述如下:

1. 若任何 MIN 结点的 β 值小于或等于任何它的先辈 MAX 结点的 α 值, 则可中止该 MIN 结点以下的搜索, 然后这个 MIN 结点的最终倒推值即为它已得到的 β 值。该值与真正的极大极小的搜索结果的倒推值可能不相同, 但是对开始结点而言, 倒推值是相同的, 使用它选择的走步也是相同的。
2. 若任何 MAX 结点的 α 值大于或等于它的 MIN 先辈结点的 β 值, 则可以中止该 MAX 结点以下的搜索, 然后这个 MAX 结点处的倒推值即为它已得到的 α 值。

当搜索用规则 1 终止时, 我们说进行了 α 剪枝, 而当搜索用规则 2 终止时, 我们说进行了

β 剪枝，保存 α 和 β 值，并且当可能的时候就进行剪枝的整个过程通常称为 $\alpha-\beta$ 过程，当初始结点的全体后继结点的最终倒推值全都给出时，上述过程便结束。在搜索深度相同的条件下，采用这个过程所获得的走步总跟简单的极大极小过程的结果是相同的，区别只在于 $\alpha-\beta$ 过程通常只用少得多的搜索便可以找到一个理想的走步。

图 2-25 给出了一个 $\alpha-\beta$ 过程的应用例子。图中结点 A、B、C、D 处都进行了剪枝，剪枝处用两横杠标出。实际上，凡减去的部分，搜索时是不产生的。

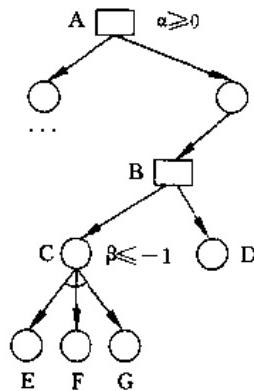


图 2-23 C 的 $\beta \leq A$

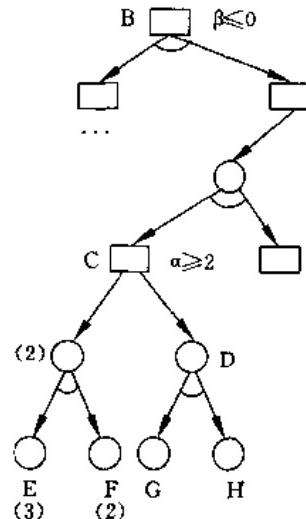


图 2-24 C 的 $\alpha \geq B$ 的 β

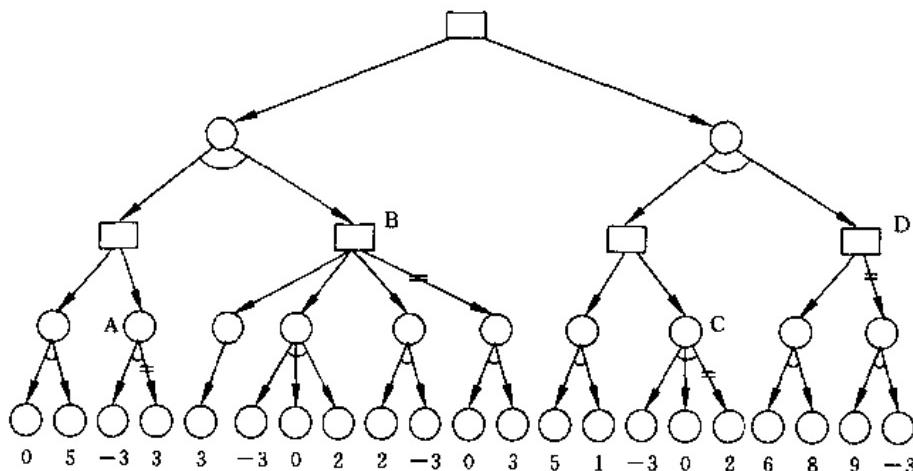


图 2-25 $\alpha-\beta$ 修剪

习题二

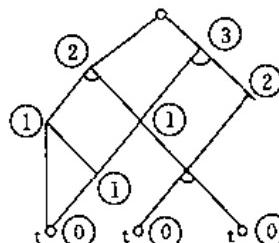
- 状态空间法和问题归约法的要点是什么？它们有何异同点？
- 设有三个传教士和三个野人来到河边，打算乘一只船从右岸渡到左岸去。该船的负载能力为两人。在任何时候，如果野人人数超过传教士人数，那么野人就会把传教士吃掉。他们怎样才能用这条船安全地把所有人都渡过河去？

3. 对于九宫重排问题,设 $h(n)$ 是棋子不在目标位置上的个数,并设初始态和目标态如下:

S_0	1 5 2	4 3	6 7 8
S_9	1 2 3	4 5	6 7 8

画出从 S_0 到 S_9 的启发式搜索图。

4. 什么是与或图的解图? 设与或图如下:



其中结点处标的是启发式函数 $h(n)$ 的值,两结点之间的耗费均设为 1,在与结点处用和耗计算。试画出求解的搜索过程。

5. 哪种问题空间“深度优先搜索”优于“广度优先搜索”? 什么情况下“广度优先搜索”又优于“深度优先搜索”?

6. A^* 和 AO^* 算法有何区别?

7. 试用四元组表示四圆盘梵塔问题,并画出求解该问题的与或图。

8. 用与或图表示“猴子和香蕉”问题。

一只猴子位子水平位置 a 处,香蕉挂在水平位置 c 处的上方,猴子想吃香蕉,但高度不够,够不着。恰好在 b 处有可移动的台子,若猴子站在台子上,就可以够到香蕉。问题是判定猴子的行动计划,使它能够到香蕉。

提示:状态用四元组描述

$$S = (W, X, Y, Z)$$

其中: W 猴子所处水平位置

X 台子所在水平位置

Y 猴子是否在台子上(在, $Y=1$; 不在, $Y=0$)

Z 猴子是否能够到香蕉(够到, $Z=1$; 没够到, $Z=0$)

允许的操作集为

$$F = \{f_1, f_2, f_3, f_4\}$$

其中: $f_1(U)$ 为猴子走到 U 处。 $(W, X, 0, Z) \rightarrow (U, X, 0, Z)$

$f_2(V)$ 为猴子推台子到 V 处。 $(X, X, 0, 0) \rightarrow (V, V, 0, 0)$

f_3 为猴子爬上台子。 $(X, X, 0, Z) \rightarrow (X, X, 1, Z)$

f_4 为猴子够到香蕉。 $(C, C, 1, 0) \rightarrow (C, C, 1, 1)$

问题的初始状态空间为:

$$\langle \{S_0\}, \{F\}, \{S_4\} \rangle$$

第三章 基于逻辑的问题求解方法

逻辑亦称为数理逻辑或符号逻辑,主要研究关于推理,证明等模拟人类智能的问题求解理论和方法,是人工智能的重要基础。逻辑可划分为经典(标准)逻辑和非经典逻辑两大类,前者包括命题逻辑和一阶谓词逻辑;后者包括模态逻辑、模糊逻辑、时序逻辑、非单调逻辑,多值逻辑等。本章主要介绍一阶谓词逻辑、时序逻辑的基本理论及其在人工智能问题求解中的应用。

第一节 一阶谓词逻辑基础

由离散数学可知,其取值为其真或假(表示是否成立)的句子称作命题,带有变量(参数)的命题称为谓词。谓词用来描述个体(可以独立存在的事物)之间的关系或属性。以谓词为基础的谓词演算是一种形式语言,可严密而精确地表达复杂的人类知识,并作为演绎推理的重要基础。

一、谓词逻辑的符号体系

在谓词逻辑使用的符号一般包括:

- 标点符号及括号
- 常量: 以小写字母组成的字符串
- 变量符号: 习惯上是小写字母 x, y, z, u, v, w
- 函数符号: 通常以小写字母或小写字母串表示
- 谓词符号: 通常以大写字母或大写字母串表示

函数与谓词的形式分别为:

谓词 $P(x_1, x_2, \dots, x_n)$

函数 $f(x_1, x_2, \dots, x_n)$

x_1, x_2, \dots, x_n 为个体变量

以上符号中常量用来表示特定的事物或概念(个体);变量表示非特定的事物或概念;谓词用来表达 n 个实体之间的关系或属性,其取值为 T (真)或 F (假);函数仅实现个体域中 n 个个体到某一个体的映射,没有真假取值。

- 连接词:

\sim : 否定(非)

\wedge : 合取(与)

\vee : 析取(或)

\rightarrow : 蕴含(IF……THEN)

\equiv : 等价(双条件)

- 量词:

\forall : 全称量词, 表示所有的。例如, 对于个体域中所有个体 x , 谓词 $F(x)$ 均成立时, 可用含有全称量词的谓词表示为

$$\forall x \ s(F(x))$$

\exists : 存在量词, 表示存在某一些。例如, 若存在某些个体 x , 使谓词 $F(x)$ 成立时, 可用含有存在量词的谓词表示为

$$\exists x \ (F(x))$$

利用上述符号, 可把单个谓词组合成复杂的谓词公式, 表达复杂的领域知识。

例 3-1 用谓词 $S(x)$ 表示个体 x 学习好, $W(x)$ 表示 x 工作好; 谓词公式 $S(x) \wedge W(x)$ 表示 x 不仅学习好而且工作好, $S(y) \wedge \neg S(y)$ 表示 y 的学习好但工作不好; 谓词公式 $\forall z \ Computer(z) \rightarrow CPU(z)$ 表示所有的计算机(个体 z)都有 CPU。

例 3-2 设有三个积木块 a, b, c , 它们之间的位置关系可用下列谓词表示:

$ON(a, b)$ 表示 a 在 b 之上;

$ON(b, c)$ 表示 b 在 c 之上;

$ON(a, b) \wedge ON(b, c) \rightarrow ON(a, c)$ 表示 a 在 b 之上且 b 在 c 之上, 则 a 在 c 之上。

若量词仅对谓词的个体(变量)而不能对谓词自身起限定作用, 即把谓词名视为常量时, 称其为一阶谓词; 若量词不仅对个体, 而且对谓词自身起限定作用, 称其为高阶谓词。例如:

$\exists P \ (Q(x) \rightarrow P), \forall Q \forall y Q(y)$ 均为二阶谓词; $\forall x \forall y P(x, y)$ 是一阶谓词。经典逻辑中最重要的几类逻辑是命题逻辑、谓词逻辑和二阶逻辑。命题逻辑表达能力弱, 能解决的问题不多; 二阶逻辑过于复杂, 且到目前为止不存在有效的算法。在人工智能中常用的还是一阶谓词逻辑, 因此本章介绍的内容主要针对一阶谓词逻辑。

二、谓词演算公式

不含任何连接词及量词的谓词公式, 是谓词演算的基本公式, 称为原子公式。

由 n 元谓词 F 及其 n 个个体变量 x_1, x_2, \dots, x_n 所构成的公式 $F(x_1, x_2, \dots, x_n)$ 是一个原子公式。在谓词演算中包含命题演算, 所以命题变量也是一个原子公式。

下面给出谓词演算的合式公式(简称公式或 WFF)的递归定义;

(1) 谓词演算的原子公式是公式。

(2) 若 A 是公式, 则 $\neg A$ 也是公式。

(3) 若 A, B 是公式, 则 $A \wedge B, A \vee B, A \rightarrow B, A \equiv B$, 也都是公式。

(4) 若 A 是公式, x 是个体变量, 则 $\forall x(A), \exists x(A)$ 也是公式。

(5) 只有按(1)–(4)所得才是公式。

三、谓词公式的解释

与命题类似, 每个谓词及公式也都有由人赋予的一定的语义(含义), 但从谓词及公式本身却无法推出其语义。因此, 在应用谓词逻辑解决问题时, 必须对谓词公式进行解释, 即人为地给谓词公式指派一定的语义。对一阶谓词公式 P 在个体域 D 上的解释包括:

• 为 P 的每个常量赋予 D 中的一个元素;

• 为 P 的每个 n 元函数指派一个 D^n 到 D 的映射;

- 为 P 的每个 n 元谓词指派一个 D^n 到 T, F 的映射。
- D^n 是 D 的元素组成的 n 元组集合。

对于一个谓词公式 P 的解释可有多种,其中一些解释可使 P 为真,而另一些解释则可使 P 为假。若在所有可能的解释下 P 均为真,称 P 为永真式,或普遍有效的公式;若仅在某一特定的解释下 P 为真,称 P 为可满足的公式;若在所有可能的解释下 P 均为假,称 P 为永假式,或不可满足的公式。但是,判断一个公式是否永真是非常困难的,特别是对任一谓词公式,在有限步内判别其是否为永真式的算法目前还未找到。

四、谓词演算的基本等价式及推理规则

若两个 WFF U 及 V 在任一解释下其值完全相同,则称这两个 WFF 等价,表示为 $U \equiv V$ 。由于谓词逻辑是命题逻辑的推广,命题逻辑中的基本等价式和推理规则在谓词逻辑仍可沿用。但由于谓词逻辑中引入了变量及量词,须再增加一些与变量与量词有关的一些定理和规则,现一并归纳于下:

- 双重否定律:

$$\sim(\sim P(x)) \equiv P(x)$$

- 摩根定律:

$$\sim(P(x) \vee Q(x)) \equiv \sim P(x) \wedge \sim Q(x)$$

$$\sim(P(x) \wedge Q(x)) \equiv \sim P(x) \vee \sim Q(x)$$

- 逆否律:

$$P(x) \rightarrow Q(x) \equiv \sim Q(x) \rightarrow \sim P(x)$$

- 分配律:

$$P(x) \wedge (Q(x) \vee R(x)) \equiv (P(x) \wedge Q(x)) \vee (P(x) \wedge R(x))$$

$$P(x) \vee (Q(x) \wedge R(x)) \equiv (P(x) \vee Q(x)) \wedge (P(x) \vee R(x))$$

- 结合律:

$$(P(x) \wedge Q(x)) \wedge R(x) \equiv P(x) \wedge (Q(x) \wedge R(x))$$

$$(P(x) \vee Q(x)) \vee R(x) \equiv P(x) \vee (Q(x) \vee R(x))$$

- 蕴含等价式:

$$P(x) \rightarrow Q(x) \equiv \sim P(x) \vee Q(x)$$

- 易名规则:

$$\forall x P(x) \vee \forall x Q(x) \equiv \forall x P(x) \vee \forall y Q(y)$$

- 量词转换律:

$$\sim \forall x P(x) \equiv \exists x \sim P(x)$$

$$\sim \exists x Q(x) \equiv \forall x \sim Q(x)$$

- 量词分配律:

$$\exists x (P(x) \vee Q(x)) \equiv \exists x P(x) \vee \exists x Q(x)$$

$$\forall x (P(x) \wedge Q(x)) \equiv \forall x P(x) \wedge \forall x Q(x)$$

$$\forall x (P \rightarrow Q(x)) \equiv P \rightarrow \forall x Q(x)$$

$$\exists x (P \rightarrow Q(x)) \equiv P \rightarrow \exists x Q(x)$$

- 量词交换律:

$$\forall x \forall y (P(x, y)) \equiv \forall y \forall x (P(x, y))$$

$$\forall x \forall y (P(x, y)) \equiv \exists y \forall x (P(x, y))$$

$$\forall x \forall y (P(x, y)) \equiv \exists x \forall y (P(x, y))$$

$$\exists y \forall x (P(x, y)) \equiv \forall x \exists y (P(x, y))$$

- 量词辖域变换等价式：

$$\forall x (P(x) \vee Q) \equiv \forall x (P(x) \vee Q)$$

$$\forall x (P(x) \wedge Q) \equiv \forall x (P(x) \wedge Q)$$

$$\exists x (P(x) \vee Q) \equiv \exists x (P(x) \vee Q)$$

$$\exists x (P(x) \wedge Q) \equiv \exists x (P(x) \wedge Q)$$

Q 中不含变量

- 量词消去及引入规则：

- 全称量词消去规则： $\forall x P(x) \equiv P(y)$

- 全称量词引入规则： $P(y) \equiv \forall x P(x)$

- 存在量词消去规则： $\exists x Q(x) \equiv Q(c)$ (c 为常量)

- 存在量词引入规则： $Q(c) \equiv \exists x Q(x)$

- 有限域量词消去规则：设有限个体域为 $D \in d_1, d_2, \dots, d_n$

$$\forall x P(x) \equiv P(d_1) \wedge P(d_2) \wedge \dots \wedge P(d_n)$$

$$\exists x Q(x) \equiv Q(d_1) \vee Q(d_2) \vee \dots \vee Q(d_n)$$

五、谓词逻辑的演绎推理方法

推理是根据一定的准则由称为前提的一些判断导出称为结论的另一些判断的思维过程。传统推理方法可以有多种方式分类：

- 按推理所依据的基本理论或原理，可分为演绎推理（根据命题及谓词逻辑理论进行推演）；归纳推理（从特殊事物或现象推出普遍性规律）；类比推理（根据相似原理在具有许多相同属性的两类事物基础上，推断出这两类事物的其他相同属性）。

- 根据对推理过程的置信程度，可分为精确性推理、不精确性推理（统计推理、模糊推理等）。

- 根据推理结论与前提的对应增长关系，可分为单调推理与非单调推理（默认推理、界限推理等）。

- 根据推理过程进行的方向，可分为正向推理、反向推理、正反向混合推理。

在传统的智能系统，特别是定理证明系统中，采用最多的是以形式逻辑为基础的演绎推理。

下面主要介绍以一阶谓词逻辑为基础进行逻辑演绎推理的基本方法和规则。

(一) 推理形式

若在使谓词公式 $P = P_1 \wedge P_2 \wedge \dots \wedge P_n$ 为真的任一解释下，均有另一谓词公式 R 为真，就说由 P 推出了 R ， P 为 R 的前提， R 为 P 的逻辑结论，用逻辑符号表示为

$$P \vdash R \text{ (或 } P \rightarrow R\text{)}.$$

换言之，

$$P \vdash r$$

当且仅当

- (1) $P_1 \wedge P_2 \wedge \cdots \wedge P_n \rightarrow R$ 是有效的；
- (2) $P_1 \wedge P_2 \wedge \cdots \wedge P_n \rightarrow \sim R$ 是不可满足的。

进行逻辑演绎推理就是使用证明公式 $P_1 \wedge P_2 \wedge \cdots \wedge P_n \rightarrow R$ 是否成立。

(二) 推理规则

- 前提及结论引入规则

在推理过程中可随时引入前提，并把推理得到的中间结论作为后继推理的前提。

- 替换规则

谓词公式中任一部分公式都可用其等价的公式替换。

- 假言推理规则

已知谓词公式 $C \rightarrow D$ 及 E , 则可推出公式 E 。

(三) 推理过程

推理过程就是反复使用谓词演算的基本等价式及推理规则，对已知谓词公式进行变换，以得到所需逻辑结论的过程。

六、谓词公式的规范化

为了方便使用 WFF 进行定理证明和逻辑推理，需要把 WFF 变换为便于使用的规范形式，称为 WFF 范式。几种典型的范式是前束范式，SKOLEM 范式及 HORE 子句。

(一) 前束范式与 SKOLEM 范式

若一个谓词公式 P 的所有量词均非否定地出现在 P 的前部，且量词辖域是整个 WFF，称 P 为前束范式或前束标准形。任一 WFF 都可化为与之等值的前束范式。消去前束范式中的所有存在量词后所得到的谓词公式称为 SKOLEM(或 S 范式)。S 范式与原式不一定等值，但与原式在不可满足性方面是等价的(即若原式是不可满足的，则对应的 S 范式也是不可满足的)。

前束范式的一般形式为

$$P = \Phi_{x_1} \Phi_{x_2} \cdots \Phi_{x_n} M(x_1, x_2, \dots, x_n)$$

式中 Φ 可以是 \forall 或 $\exists E$, M 是 P 的母式, M 中不含有量词。前束范式的两个例子如下：

$$\forall x \forall y (P(x, y) \wedge Q(x, y))$$

$$\forall x \forall y \exists z (V(x, y) \rightarrow \sim W(z))$$

在把一个 WFF 化为前束范式及 S 范式的过程中，须反复运用谓算的基本定理和等价式，直到 WFF 满足范式的要求。化 WFF 为前束范式的基本步骤如下：

1. 消去可能出现的多余量词(在母式 M 中无相应变量的量词)；
2. 利用蕴含等价式消去蕴含连接词 \rightarrow ；
3. 利用摩根定律内移否定词 \sim 的辖域范围，使其仅作用于原子公式；
4. 在一量词辖域内，受该量词约束的变量可为任一未出现过的变量名所替换而不改变原式的真值。根据这个性质，可对变量作必要的换名，使式中每一变量词仅受一个量词约束；
5. 利用量词辖域变换律把量词的辖域范围扩至整个 WFF，得到一个前束范式；

6. 消去式中的存在量词,把所得前束范式化为 S 范式。设 Ψ 是前束范式 PF 中的存在量词,若 Ψ 之前无全称量词,可用一个在 M 中未出现过的常量 c 代替 Ψ 眩域中的所有变量,并消去 Ψ ;若 Ψ 前有全称量词,则可用一个与 Ψ 的约束变量有关的函数(称为 SKOLEM 函数)替代 Ψ 眩域中的所有变量,并消去 Ψ 。

7. 利用结合律、分配律等把母式 M 化为合取范式(即用符号 \wedge 连接的公式,其中每个合取项均为原子公式)。

经过上述 7 个步骤,就把 WFF 变换成了一个以合取式为母体的 S 范式。

(二)谓词子句

在谓词演算中另一种范式称为子句,它是比 S 范式更小,更为实用的一个知识单位。我们已经知道,不含任何连接词的谓词公式是原子公式,称原子公式及其否定为文字;一些文字的析取式(或项)称为子句;一些子句的集合称为子句集。由于 WFF 的 S 范式已经是消去了存在量词的合取范式,其母式的每一个合取项都是一个子句,换言之,S 范式的母式由数个子句的合取所组成。因为 S 范式母式 M 中的变量均受前面全称量词的约束,去掉这些量词导致发生误解,故可隐去 M 前面的量词。隐去 S 范式前部的全称量词,并以逗号替代式中所有的合取符号,就可得到原 WFF 对应的子句集。

例 3-3 把 WFF

$$G = \forall x P(x) \rightarrow \forall y [P(y) \rightarrow P(f(x, y))] \wedge \sim (\forall y [Q(x, y) \rightarrow P(y)])$$

变换为子句集,过程如下:

(1)重复运用蕴含等价式,消去蕴含词

$$G = \forall x \sim P(x) \vee \forall y [\sim P(y) \vee P(f(x, y))] \wedge \sim (\forall y [\sim Q(x, y) \vee P(y)])$$

(2)重复运用摩根定律内移否定词的辖域范围

$$\begin{aligned} G &= \forall x \sim P(x) \vee \forall y [\sim P(y) \vee P(f(x, y))] \wedge \exists y \sim [\sim Q(x, y) \vee P(y)] \\ &= \forall x \sim P(x) \vee \forall y [\sim P(y) \vee P(f(x, y))] \wedge \exists y [Q(x, y) \wedge \sim P(y)] \end{aligned}$$

(3)变量改名

$$G = \forall x \sim P(x) \vee \forall y [\sim P(y) \vee P(f(x, y))] \wedge \exists w [Q(x, w) \wedge \sim P(w)]$$

(4)移所有全称量词到 M 前部,形成前束范式

$$G = \forall x \forall y \sim P(x) \vee [\sim P(y) \vee P(f(x, y))] \wedge \exists w [Q(x, w) \wedge \sim P(w)]$$

(5)消去存在量词,形成 G 对应的 S 范式

$$\forall x \forall y \sim P(x) \vee [\sim P(y) \vee P(f(x, y))] \wedge [Q(x, g(x)) \wedge \sim P(g(x))]$$

$g(x)$ 为设定的 SKOLEM 函数

(6)化母式 M 化为合取范式

$$\begin{aligned} \forall x \forall y [\sim P(x) \vee P(y) \vee P(f(x, y))] \wedge [\sim P(x) \vee Q(x, g(x))] \\ \wedge [\sim P(x) \vee \sim P(g(x))] \end{aligned}$$

(7)隐去 M 前面的量词,得

$$\begin{aligned} [\sim P(x) \vee P(y) \vee P(f(x, y))] \wedge [\sim P(x) \vee Q(x, g(x))] \\ \wedge [\sim P(x) \vee \sim P(g(x))] \end{aligned}$$

(8)以逗号替代所有的合取符号,得到由 3 个子句组成的子句集

$$\begin{aligned} \sim P(x) \vee P(y) \vee P(f(x, y)), \\ \sim P(x) \vee Q(x, g(x)), \\ \sim P(x) \vee \sim P(g(x)), \end{aligned}$$

$$\sim P(x) \vee \sim P(g(x))$$

(三)HORN 子句

最多含一个非否定原子公式(又叫为正文字)的谓词子句称为 HORN 子句,其一般形式为:

$$\sim A_1 \vee \sim A_2 \vee \sim A_3 \cdots \vee \sim A_n \vee B$$

此式等价于

$$A_1 \wedge A_2 \wedge A_3 \cdots \wedge A_n \rightarrow B$$

式中 A_i, B 均为谓词子句, $i \in 1, 2, \dots, n$

可见,HORN 子句是包含若干个条件(前提)子句,仅含一个结论子句的谓词公式。凡是可用一阶逻辑描述的问题,均可以 HORN 子句来表达。由于运用 HORN 子句表达知识方便,推理比较简单,已成为人工智能问题求解的重要工具。在以一阶逻辑为基础的 PROLOG 语言中就应用了 HORN 子句。

七、置换与合一

置换是指用特定的变量 v_i 替代公式中的某些项 t_i ,可一般地表示为有限集:

$$t_1/v_1, t_2/v_2, \dots, t_n/v_i, \dots, t_n/v_n$$

其中 v_i 是变量, t_i 是异于 v_i 的项; $v_i \neq v_j, i \neq j, i, j \in 1, 2, \dots, n$

对公式进行置换后得到的新公式称为公式的实例。

例 3-4 对公式 $P[x, f(y), B]$ 施加如下四个置换:

$$R_1 = z/x, w/y$$

$$R_2 = A/y$$

$$R_3 = g(z)/x, A/y$$

$$R_4 = c/x, A/y$$

得到四个公式实例:

$$P[z, f(w), B]$$

$$P[x, f(A), B]$$

$$P[g(z), f(A), B]$$

$$P[c, f(A), B]$$

合一是通过对项进行置换,使两个公式一致的过程,通过合一,可把若干个公式合为一个公式。合一的一般定义为:

设有公式集 $F = F_1, F_2, \dots, F_n$, 置换 Φ ,使得

$$F_1//\Phi = F_2//\Phi = \dots = F_n//\Phi$$

则称 F_1, F_2, \dots, F_n 是可合一的,且称 Φ 为合一置换。符号串 $F_*//\Phi$ 表示对公式 F_* 施加 Φ 置换。显而易见, Φ 的作用是使集合 F 合为一个公式。

例如,公式集 $P[x, f(y), B], P[x, f(B), B]$ 的合一置换为 $\Phi = A/x, B/y$,这是因为 $P[x, f(y), B]//\Phi = P[x, f(B), B]//\Phi = P[A, f(B), B]$ 。

对公式集的合一置换不是唯一的,为了方便推理,通常还须寻找一个限制最少,最一般(通用)的合一置换 MGU(Most General Unifier)。MGU 的定义是:

若 Ψ 是公式集 F 的一个合一置换,且对 F 的任一合一置换 Φ 都存在一个置换 Ω ,使 $\Phi=\Psi\odot\Omega$ ($\Psi\odot\Omega$ 意为 Ψ 和 Ω 的复合代换),则称 Ψ 是 F 的一个 MGU。

第二节 基于一阶谓词逻辑的知识表达

谓词逻辑的表现方式与人类自然语言比较接近,能够自然而精确地表达人类思维和推理的有关知识,因此,谓词逻辑已成为各种智能系统中最基本的知识表达方法。有关谓词逻辑的基础前面已经详述,此处将使用一阶谓词逻辑的知识表达法(PBKR)用 BNF(Backus Normal Form)归纳描述如下:

```
<知识> ::= {<谓词表达式>}
<谓词表达式> ::= <蕴含式> | [量词]<(变量){, (变量)}><(蕴含式)>
<蕴含式> ::= <或式> | <或式> → <或式>
<或式> ::= <与式> | <与式> ∨ <与式>
<与式> ::= <文字> | <与式> ∧ <文字>
<文字> ::= <原子> | ~<原子>
<原子> ::= <谓词>[<(变量){, (变量)}>] | <谓词>[<(常量){, (常量)}>]
<量词> ::= (A ) | (E )
<变量> ::= <变量名>
<常量> ::= <个体名> | <常数>
```

用 PBKR 表达知识,建造知识库的基本步骤如下:

- (1) 获取并理解领域知识;
- (2) 用自然语句描述领域知识;
- (3) 使用适当的谓词公式表达自然语句;
- (4) 构造 WFF,使其与被表达的自然语句在逻辑上保持一致。

用 PBKR 法表达知识的优点是表达严密、精确、推导可以形式化、机械化。其缺点是具有单调性,不适用于动态变化环境的应用;工作效率低,实际知识的表达过分冗长,以此为基础的逻辑演绎系统容易出现组合爆炸情况。所以在实时智能系统中,须与其他知识表示方法综合使用。

第三节 归结(消解)原理

一、归结原理简介

1965 年 Robinson 发现的归结原理曾被认为是定理证明机械化方面的一个突破,是以逻辑演绎为基础进行逻辑推理的最实用方法。它的基本思想可用如下定理解释:

定理 3-1 设给定谓词公式 G 对应的子句为 S ,当且仅当 S 是不可满足的, G 也是不可满足的。

由定理可知,尽管 G 与 S 可能不等值,但在不可满足的意义下,它们是等价的。若要证明 $G=A \rightarrow B$ 成立,只要证明 $G=A \wedge \neg B$ 对应的子句集是不可满足的即可。因此,为了证明一个

难以直接推证的复杂定理,可把已知条件表示谓词公式,再化为一组子句;欲证的目标也化为一个子句的否定,把目标子句与条件子句合在一起推理,若能推出矛盾(空子句),则可证明此目标是成立的。

一般地,设谓词公式

$$G = A_1 \wedge A_2 \wedge \cdots \wedge A_n \rightarrow B$$

可化为子句集

$$S = S_1, S_2, \dots, S_n$$

对于 S 的任意两个子句 $LF1 = C_1 \vee \alpha$, $LF2 = C_2 \vee \beta$, 其中 C_1, C_2 是具有相同谓词名,但不含相同变量的两个互补原子公式(文字),若 C_1, C_2 有最一般的合一置换 Φ ,则可通过归结,即通过对 $LF1, LF2$ 进行析取运算,消去(删除)互补对,而产生一个新子句 $LNEW = (\alpha \vee \beta) // \Phi$,称 $LNEW$ 为归结式, C_1, C_2 为归结的文字。已经证明,两子句的归结式是它们的逻辑结论。把归结式 $LNEW$ 加进 S 中,构成新子句集 S' 。对 S' 重复以上归结过程,反复进行置换与合一,搜索互补文字,寻求归结式,直到 S' 中出现空子句,归结过程停止,表明 S 对应的谓词公式是不可满足的,从而与 S 对应的谓词公式 P 是成立的。

可见,实现归结推理的关键步骤是置换合一与寻求子句集中的互补子句对,由于寻求互补子句对的方式不是唯一的,所产生的归结式也不是唯一的。

以上归结原理可用假语言简要描述如下:

归结过程:

begin

化基本公式为子句集 S ;

while S 中没有空子句 DO

begin

在 S 中选取两个可归结子句 c_i, c_j ;

计算 c_i, c_j 的归结式 $f_{i,j}$;

$S = S \cup f_{i,j}$;

end

end

例 3-5 已有下列公式,求证目标 G 是 F_1, F_2 的逻辑结论

$$F_1: \forall x(C(x) \rightarrow (W(x) \wedge R(x)))$$

$$F_2: \exists x(C(x) \wedge O(x))$$

$$G: \exists x(O(x) \wedge R(x))$$

证:先将 $F_1, F_2, \neg G$ 化为 S 范式,得如下子句集元素:

由 F_1 得: (1) $\neg C(x) \vee W(x)$

(2) $\neg C(x) \vee R(x)$

由 F_2 得: (3) $C(a)$

(4) $O(a)$ (a 为引入的任意常数)

由 G 得: (5) $\neg O(x) \vee \neg R(x)$

对(2)施加置换 a/x ,再与(2)归结,得

(6) $R(x)$

(5), (6) 归结,得(7) $\neg O(x)$

对(4)施加置换 a/x , 再与(7)归结, 得

$$(8) \sim O(x) \vee O(x) = \square$$

所以, 目标 G 是, F_1, F_2 的逻辑结论。

已经证明, 归结推理方法是完备的, 即若定理是成立的, 使用归结方法必定能得到证明。归结推理的完备性定理如下:

定理: 子句集 S 是不可满足的, 当且仅当存在一个使用归结推理方法从 S 到空子句的推理过程。

二、归结原理在问题求解中的应用举例

应用归结推理方法进行问题求解时, 要先把问题转化为对公式

$$S \rightarrow L \quad (S \text{ 为求解问题对应的公式集}, L \text{ 为求解目标})$$

的证明, 然后通过下述归结反向推理过程完成证明:

- (1) 否定 L , 得 $\sim L$;
- (2) 把 $\sim L$ 添加到 S 中, 得新集合 $S, \sim L$;
- (3) 把 $S, \sim L$ 化为子句集;
- (4) 重复归结过程, 直至导出一个表示矛盾的空子句。

例 3-6 已知每个使用 Internet 网络的人都想从网络获得信息, 证明如网络没有信息就不会有人使用 Internet。

证: 设 $U(x, y)$ 表示 x 使用 y ;

$E(u, v)$ 表示 u 得到 v ;

$I(z)$ 表示 z 是 Internet;

$F(u)$ 表示 u 是信息;

前提: $F = \forall x [\exists y (U(x, y) \wedge I(y)) \rightarrow \exists u (F(u) \wedge E(x, u))]$

结论: $G = \sim \exists u F(u) \rightarrow \forall x \forall y (I(y) \rightarrow \sim U(x, y))$

化前提为 S 范式

$$\begin{aligned} F &= \forall x [\sim (\exists y U(x, y) \wedge I(y)) \vee \exists u (F(u) \wedge E(x, u))] \\ &= \forall x [\forall y (\sim (U(x, y) \wedge I(y))) \vee \exists u (F(u) \wedge E(x, u))] \\ &= \forall x [\forall y (\sim U(x, y) \vee \sim I(y)) \vee \exists u (F(u) \wedge E(x, u))] \end{aligned}$$

引入 SKOLEM 函数 $u = f(x)$, 消去 $\exists u$, 得

$$F' = \forall x [\forall y (\sim U(x, y) \vee \sim I(y)) \vee (F(f(x)) \wedge E(x, f(x)))]$$

隐去全称量词, 得到 F 对应子句集的两个子句:

- (1) $\sim U(x, y) \vee \sim I(y) \vee (F(f(x))$
- (2) $\sim U(x, y) \vee \sim I(y) \vee E(x, f(x))$

否定结论 G , 并化为 S 范式

$$\begin{aligned} \sim G &= \sim [\sim \exists u F(u) \rightarrow \forall x \forall y (I(y) \rightarrow \sim U(x, y))] \\ &= \sim [\exists u F(u) \vee \forall x \forall y (\sim I(y) \vee \sim U(x, y))] \\ &= \sim \exists u F(u) \wedge \sim [\forall x \forall y (\sim I(y) \vee \sim U(x, y))] \\ &= \forall x \forall y \sim \exists u F(u) \wedge I(y) \wedge U(x, y) \end{aligned}$$

引入常量 c , 消去 $\sim G$ 中的存在量词, 得

$$\forall x \forall y \sim F(c) \wedge I(y) \wedge U(x, y)$$

隐去全称量词, 得到 G 对应子句集的三个子句:

$$(3) \sim F(c)$$

$$(4) I(y)$$

$$(5) U(x, y)$$

对(1)施加置换 $f(x)/c$, 再与(3)归结, 得

$$(6) \sim U(x, y) \vee \sim I(y)$$

由(5), (6)归结, 得

$$(7) \sim I(y)$$

(4), (7)归结, 得

$$\sim I(y) \vee I(y) = \square$$

故, 结论 G 得证。

归结原理等逻辑演绎推理以严密的逻辑公理系统为基础, 其优点是自然、严密, 便于推理机械化, 但求解复杂的现实问题时, 搜索工作量太大, 效率低, 可能出现组合爆炸, 因而主要用于机器定理证明。在各种应用智能系统, 如专家系统中使用的大多是基于规则的条件检索、执行型的演绎推理。

第四节 基于规则的演绎推理

在基于规则的演绎推理系统中, 有两种不同形式的谓词公式, 即事实和规则。事实是不含蕴含符号的谓词公式, 表示与求解问题有关的客观情况, 证据等。一条规则就是一个蕴含式:

$$A_1 \wedge A_2 \wedge \dots \wedge A_n \rightarrow B$$

它实际上是一组条件……结论(*IF... THEN*)表达式

$$IF \quad A_1 \wedge A_2 \wedge \dots \wedge A_n \quad THEN \quad B$$

其中, A_1, A_2, \dots, A_n 是表示前提条件的谓词公式, B 是表示结论的谓词公式。

多条规则可构成一个规则库。基于规则的演绎推理系统是一个直接证明系统, 它不像归结推理那样把蕴含式化为子句集再进行反演, 而是直接使用事实及蕴含规则推理, 这样可以避免在从蕴含式到子句集转化过程中, 包含在蕴含式中的一些有用的启发式信息的丢失, 同时使知识表达及推理过程更加直观, 易于为人们所接受。

一、基于规则的正向演绎推理(FR)

FR 系统的事实集构成了系统的初始全局数据库, 表示求解问题的初始状态及有关信息。表示事实的谓词公式集被转化为一系列与或表达式(即由 \wedge, \vee 连接的一些原子公式)。转化步骤是先把事实谓词公式化为 S 范式, 再隐去式中的全称量词(注意, 不需要像子句那样转换为合取范式)。

可用一棵与或树表示一个与或表达式, 树中根结点代表整个表达式, 叶子结点表示式中不可再分解的原子公式, 其他结点表示尚可分解的子表达式。

FR 系统的规则称为 F 规则, 用来控制推理过程, 对事实与或树进行变换, 以求与目标表达式匹配。为了匹配方便, F 规则被限定为单前件蕴含式, 即:

$$L \rightarrow W$$

式中, L 为单文字(原子公式), W 为与或表达式, 对于简单的非单文字规则, 如

$$(L_1 \vee L_2) \rightarrow W$$

可化为两条规则

$$L_1 \rightarrow W, L_2 \rightarrow W$$

一般地, 先把非单文字规则化为 S 范式, 再隐去全称量词, 最后再恢复其蕴含符号, 变成单前件的形式。

FR 系统中的目标限定为子句形式, 这是因为由 F 规则所推导与或树的最终解图, 即叶子结点集对应于一个子句集。

FR 又称为数据驱动推理, 它从一组事实(初始与或树)出发, 尝试匹配所有可利用的 F 规则, 并不断加入新的事实到全局数据库, 对事实与或树进行合一, 替换等推导变换, 直到与或树包含目标表达式, 即与或树的叶子结点集与目标子句集匹配, 获得问题的解答。

例 3-7 已知如下的事实, 规则及目标, 用正向演绎推理求解此问题:

事实: ZML 既会用计算机又精于设计计算机, 否则 ZML 就不是计算机专家;

规则 R_1 : 所有精通计算机设计的人都是计算机专家;

R_2 : 任何只会用计算机的人是计算机用户;

目标: 有些人并不精通计算机设计, 而是计算机用户。

解: 用谓词公式表达所给事实, 规则及目标如下:

事实: $\neg \text{Expert}(zml) \vee (\text{Usege}(zml) \wedge \text{Design}(zml))$

规则 $R1$: $\forall x \text{ Proficient}(x) \rightarrow \text{Expert}(x)$

$R2$: $\forall y (\text{usege}(y) \rightarrow \text{Consumer}(y))$

目标: $\exists z (\neg \text{Proficient}(z) \vee \text{Consumer}(z))$

求解此问题就是要证明所给目标谓词公式成立, 应用前述 FR 推理方法得到的推理与或图如图 3-1 所示, 其中粗线框表示匹配所得目标结点。

二、基于规则的反向演绎推理(BR)

BR 系统的目标集构成了初始全局数据库, 表示求解问题的初始状态及有关信息。表示目标的谓词公式集被转化为一系列与或表达式, 转化步骤与 FR 系统类似。

BR 系统的规则称为 B 规则, 用来控制推理过程, 对目标与或树进行变换, 以求与事实表达式匹配。B 规则亦被限定为单后件蕴含式, 即:

$$W \rightarrow L$$

式中, 后件 L 为单文字(原子公式), 前件 W 为任意与或表达式。

BR 系统的事实被限定为文字合取式。

BR 又称为目标驱动推理, 它从一个目标表达式出发, 经反向推理链导出已知事实集。在推理过程中, 反复使用 B 规则对目标与或树逆向进行合一, 替换等推导变换, 直到与或树包含有事实结点, 即目标与或树与事实节点相匹配, 获得问题的解答。

例 3-8 已知

事实 $F_1: \text{Dog}(\text{sid})$ (狗名叫 sid)
 $F_2: \neg \text{Barks}(\text{sid})$ (sid 不叫)

	$F_3: \text{Tail}(\text{sid})$	(sid 摆尾巴)
	$F_4: \text{Scat}(\text{mik})$	(mik 是小猫)
规则	$R_1: \text{Dog}(x) \wedge \text{Tail}(x) \rightarrow \text{Kind}(x)$	(摇尾巴的狗温顺)
	$R_2: \text{Kind}(s) \wedge \neg \text{Barks}(s) \rightarrow \text{Afraid}(z, s)$	(温顺又不叫的东西不用怕)
	$R_3: \text{Dog}(u) \rightarrow \text{Animal}(u)$	(狗是动物)
	$R_4: \text{Cat}(v) \rightarrow \text{Animal}(v)$	(猫是动物)
	$R_5: \text{Scat}(w) \rightarrow \text{Cat}(w)$	(小猫是猫)

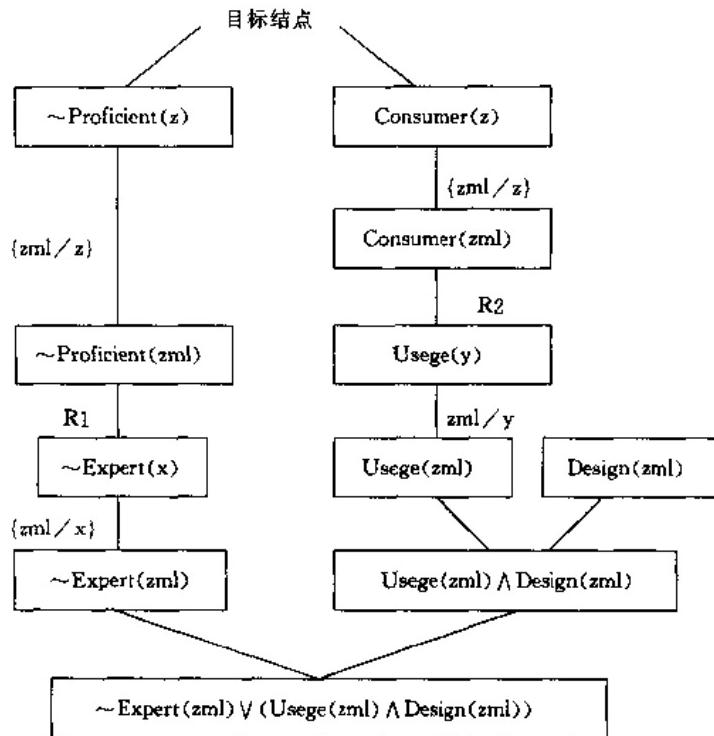


图 3-1 例 3.7 的与或图

问题：是否有一只不怕狗的猫？用反向演绎推理求解此问题。

解：所求解问题可表达为目标

$$\exists x \exists y [\text{Cat}(x) \wedge \text{Dog}(y) \wedge \neg \text{Afraid}(x, y)]$$

应用 BR 方法求解此问题的与或树图如图 3-2 所示，其中粗线框表示匹配所得事实结点。

第五节 时序逻辑

一、时序逻辑概述

时序逻辑是处理与时间有关的推理问题的一类逻辑。在时序逻辑中，相同句子在不同时刻有不同的值。在过去某时刻为真的句子可能现在不为真，现在为假的句子到将来某时刻又可能为真。为了处理这种情况，可在任一命题之前加四种时序算子 F, P, G, H 就构成了命题时序语言 lt 。 lt 中的时序算子的直觉解释如下：

FA: A 在将来某时间为真;

PA: A 在过去某时间为真;

GA: A 在将来永远为真;

HA: A 在过去永远为真。

在此基础上,引进一个时序框架 tf :

$$tf = (T, R, h)$$

是一个三元组,其中:

T 是非空的时间点集合; R 是时序优先关系; h 是一个函数;

$h: T \times lt$ 的原子子句 $\rightarrow 1, 0$

lt 是扩充的命题演算语言,其语义可通过对函数 h 的赋值来解释:

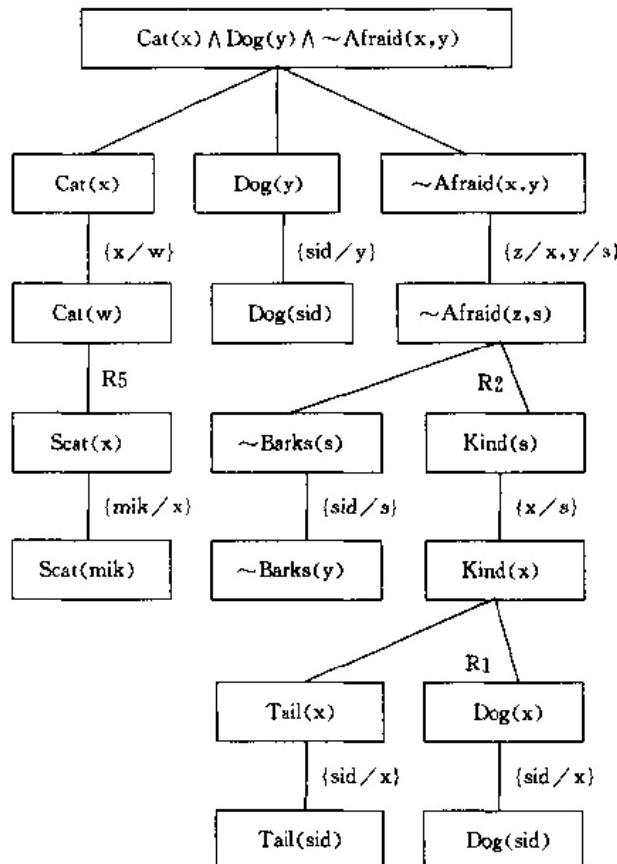


图 3-2 反向求解例 3.8 的与或树图

(1) $h(t, A \& B) = 1$, 当且仅当 $h(t, A) = 1$, 同时 $h(t, B) = 1$

(2) $h(t, \sim A) = 1$, 当且仅当 $h(t, A) = 0$

(3) $h(t, FA) = 1$, 当且仅当 $(\exists t')(R(t, t') \& h(t', A) = 1)$

(4) $h(t, PA) = 1$, 当且仅当 $(\exists t')(R(t', t) \& h(t', A) = 1)$

根据对时序优先关系 R 的限制的不同,可得到各种不同的时序逻辑。

再从对时间的表示来看,有基于时间点的时序逻辑; 基于区间的时序逻辑; 基于事件的时序逻辑等。

Allen 1981 年把两个时间点之间的一个时间区间,定义为时间原语,并规定了两个时间区

间之间的十三种不同关系。为了进行时序推理,他定义了五种区间之间的时序关系:

$\text{During}(i_1, i_2)$ 意为时区 i_1 完全包含在时区 i_2 之中,它们可在端点重叠。

$\text{Before}(i_1, i_2)$ 意为时区 i_1 在时区 i_2 之前,且无重叠。

$\text{Overlap}(i_1, i_2)$ 意为时区 i_1 在时区 i_2 之前开始,但它们有重叠。

$\text{Meets}(i_1, i_2)$ 意为时区 i_1 在时区 i_2 之前,但它们之间无时区。

$\text{equal}(i_1, i_2)$ 意为 i_1 和 i_2 是相同的时区。

这些时序关系可用图 3-3~图 3-7 来表示。

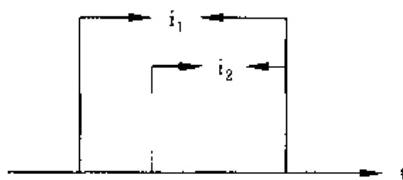


图 3-3 $\text{During}(i_1, i_2)$

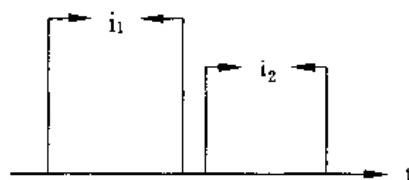


图 3-4 $\text{Before}(i_1, i_2)$

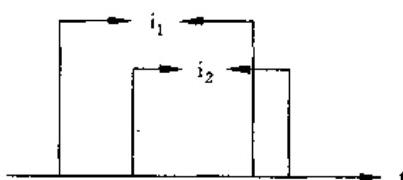


图 3-5 $\text{Overlap}(i_1, i_2)$

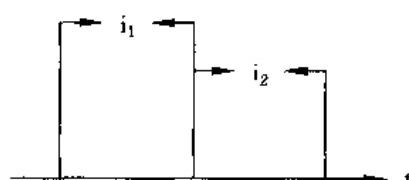


图 3-6 $\text{Meets}(i_1, i_2)$

Allen 还引入谓词 $\text{Holds}(p, i)$

表示性质 p 在区间 i 成立;引进事件和动作等概念,用谓词 $\text{Occur}(e, i)$ 表示事件 e 在区间 i 发生。

McDemott 的方法基于时间点,其中每个事件有一个时间点的起点和终点。他用谓词 $T(t, p)$ 表示事实 p 在时刻 t 成立;
 $TT(t_1, t_2, p)$ 表示事实 p 在区间 $[t_1, t_2]$ 的每一时刻成立;
 $OCC(t_1, t_2, e)$ 表示事件 e 在区间 $[t_1, t_2]$ 内发生。

Shoham 归纳了 Allen 和 McDemott 的时序逻辑表示与推理的共性,用公式

$$\text{TURE}(t_1, t_2, p)$$

表示谓词 p 在区间 $[t_1, t_2]$ 成立,并讨论了有关的时序逻辑运算等。

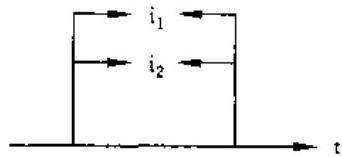


图 3-7 $\text{Equal}(i_1, i_2)$

二、基于系统状态的时序逻辑

定义 3-1 一个系统状态是过程或对象内部状态的反映,可通过系统状态变量在某段时间(时刻)的取值或改变来表征。这可用 BNF(巴克斯范式)详细描述如下:

$\langle \text{状态} \rangle ::= \langle \text{表达式} \rangle$

$\langle \text{表达式} \rangle ::= \langle \text{变量名} \rangle | \langle \text{运算符} \rangle \langle \text{值} \rangle [\langle \text{运算符} \rangle \langle \text{值} \rangle]$

$\langle \text{变量名} \rangle ::= \text{以大写字母开头的字符串}$

$\langle \text{运算符} \rangle ::= + | - | * | / | > | < | = | / | ==$

$\langle \text{值} \rangle ::= \langle \text{整数} \rangle | \langle \text{实数} \rangle | \langle \text{字符} \rangle | \langle \text{字符串} \rangle$

其中,* 表示乘,/表示除,>= 表示大于等于,<= 表示小于等于,=/表示不等于,== 表示

等于。

定义 3-2 一个系统状态结构是一个代数系统 $E=(V,A)$ 。

其中, V 是系统状态的非空集合

A 是时序关系谓词的非空集合

时序关系谓词包括:

(1) $\text{delay}(n)$

当且仅当系统延迟 n 个单位时间时为真。

(2) $\text{begin}(S,t)$

当且仅当系统状态 S 在时刻 t 发生时为真。

(3) $\text{end}(S,t)$

当且仅当系统状态 S 在时刻 t 结束时为真。

(4) $\text{last}(S,t)$

当且仅当系统状态 S 延续 n 个单位时间时为真。

(5) $\text{appear}(S_1,n,S_2)$

当且仅当系统状态 S_1 发生, 经过 n 个单位时间, S_2 才发生时为真。

(6) $\text{intr}(E,t)$

当且仅当突发事件(如控制台中断, 断电警报, 火灾警报等)在时刻 t 发生时为真。

引理 3.1 $\text{begin}(S,t_1)\text{and end}(S,t_2)$

$\rightarrow \text{last}(S,(t_2-t_1))$

引理 3.2 $\text{appear}(S_1,n_1,S_2)\text{and appear}(S_2,n_2,S_3)$

$\rightarrow \text{appear}(S_1,(n_1+n_2),S_3)$

引理 3.3 $\text{appear}(S_1,t,S_2)$

$\rightarrow \text{begin}(S_1,t)\text{and delay}(t)\text{and begin}(S_1,(t_1+t_2))$

引理 3.4 $\text{appear}(S_1,0,S_2)$

$\rightarrow \text{begin}(S_1,t)\text{and begin}(S_2,t)$

例 3-9 $\text{begin}("V50>400V",8:1:1)$

表示变量 V50(第 50 号传感器输出)值在 8 时 1 分 1 秒开始大于 400V。

$\text{appear}("T1>500C",5s,"V20 > 35V")$

表示温度变量 T1 值大于 500℃后经过 5 秒钟, 电压变量 V20 的值大于 35V。

$\text{intr}("控制台中断",13:1:0)$

表示在 3 时 1 分发生了控制台中断。

上述基于系统状态的时序逻辑本质上是嵌入了时间因素的一阶谓词逻辑, 是基于点的、基于区间的、基于事件的三种时序逻辑表示方法的综合运用。由于它允许把反映动态系统状态的运算表达式及其相关时间作为变量嵌入时序谓词, 而时序谓词又可直接作为产生式规则的前件, 这就为简捷、直观地表达复杂时变领域知识奠定了良好的基础。

为了避免经典的时序逻辑由于依赖推理规则及逻辑演绎而导致的巨大的复杂性和低效率, 作者把此处提出的基于系统状态的时序逻辑仅用作智能系统的一种外部知识表达方法, 推理将通过神经网络(参见第十五章)的并行处理而进行。这样, 既能克服经典的时序逻辑效率低的缺点, 又能发挥了它直观表达时序知识的长处。

三、基于时序规则的控制领域知识表达——时序规则法

时序规则法是作者提出的一种表达控制领域知识的新方法，简述如下：

定义 3.3 事实是一个三元组

$F(\text{事实名}, \text{属性}, \text{值})$

它反映了对已知对象属性值的描述，可用于设定一些固定不变的控制系统参数。

定义 3.4 时序规则是一个七元组

$R(\text{规则名}, \text{优先级}, \text{前件集}, \text{后件集}, \text{逻辑运算集}, \text{时序关系谓词集}, \text{注释})$

可用于描述控制系统中若干对象属性间的关系及控制规律。

时序规则的基本形式是嵌套着时序关系谓词的扩展产生式规则，以 BNF 详细表达如下：

$\langle\text{时序规则}\rangle ::= [\text{note}(\text{注释})]$

 rule⟨规则名⟩

 Priority⟨规则优先级⟩

 if

 ⟨前件⟩

 then

 ⟨后件⟩

⟨规则名⟩ ::= ⟨符号串⟩

⟨规则优先级⟩ ::= ⟨整数⟩

⟨前件⟩ ::= ⟨时序关系谓词⟩ | ⟨状态⟩

⟨时序关系谓词⟩ ::= ⟨谓词名⟩⟨参数表⟩

⟨谓词名⟩ ::= begin | end | last | delay | appear | intr

⟨参数表⟩ ::= ⟨状态⟩, [⟨时间⟩] | ⟨空⟩

⟨状态⟩ ::= ⟨表达式⟩

⟨时间⟩ ::= ⟨时:分:秒⟩ | ⟨秒⟩

⟨后件⟩ ::= ⟨动作⟩ | ⟨结论⟩ [⟨可信度⟩]

⟨动作⟩ ::= ⟨赋值表达式⟩ | ⟨命令⟩

⟨赋值表达式⟩ ::= ⟨变量名⟩ = ⟨表达式⟩

⟨命令⟩ ::= send(⟨I/O 单元名⟩) | show(⟨信息串⟩)

 | call(⟨可执行程序名⟩) | shutdown

⟨I/O 单元名⟩ ::= printer | sport | pport | esinterface

⟨结论⟩ ::= ⟨符号串⟩ [⟨符号串⟩]

⟨可信度⟩ ::= [0,1] 间的数

命令定义中的 send 表示发送信息到指定 I/O 端口；show 表示在屏幕上显示给定的信息串；call 表示调用系统外部的可执行程序；shutdown 表示关机。

I/O 单元名定义中的 printer 表示打印机；sport 表示串行口；pport 表示并行口；esinterface 表示系统到控制器的硬件接口。

表达式的定义前已给出，这里不再重复。

例 3-10 裂解炉是乙烯生产的主要设备之一，在乙烯生产过程中，必须对裂解炉进行控

制。其控制变量较多且大多与时间有关,如炉管出口温度(OCT)、进料量、稀释蒸汽流量、烧嘴压力等的控制均在一定的时间条件下进行。因此,有关裂解炉控制的知识是包含时间因素的知识,可用本节提出的时序规则法表达。

在裂解炉控制中,如 OCT 出现异常高温,就要执行如下的切断烧嘴的控制动作:

1. 如果

COT 控制回路中

闭环回路测量值大于设定值 +50℃,持续 8 秒钟;

而且开环回路测量值大于 900℃,持续 8 秒钟

则 切断 5 号烧嘴;

进行操作指导(OPG);

进行参考打印。

2. 如果

切断 5 号烧嘴后,又经过 2 分钟,在 COT 控制回路中,

闭环回路测量值大于设定值 +50℃,持续 8 秒钟;

而且开环回路测量值大于 900℃,持续 8 秒钟

则 切断 3 号烧嘴。

3. 如果

在 COT 控制回路中

闭环回路测量值大于设定值 +150℃,持续 8 秒钟;

而且开环回路测量值大于 1000℃,持续 8 秒钟

则 切断 3 号烧嘴;

进行操作指导(OPG);

进行参考打印。

这些控制动作可用时序规则法表达如下:

note: 变量 close_t——闭环回路温度测量值

note: 变量 open_t——开环回路温度测量值

note: 变量 set_t——设定值

note: 变量 fm5_status——5 号烧嘴状态,on 表示接通;off 表示切断

note: s 表示秒,m 表示分

rule:r1

priority:2

if

last("close_t>(set_t+50)",8s)

and last("open_t>900",8s)

then

send(esinterface,"切断 5 号烧嘴")

and send(esinterface,"进行 OPG")

and send(esinterface,"进行参考打印")

note: 切断 5 号烧嘴

rule: r2

priority: 1

if

```
fm5_status == off, delay(2m),
last("close_t > (set_t + 50)", 8s)
and last("open_t > 900", 8s)
```

then

```
send(esinterface,"切断 3 号烧嘴")
and send(esinterface,"进行 OPG")
and send(esinterface,"进行参考打印")
```

note: 切断 3 号烧嘴

rule: r3

priority: 1

if

```
last("close_t > (set_t + 150)", 8s)
and last("open_t > 1000", 8s)
```

then

```
send(esinterface,"切断 3 号烧嘴")
and send(esinterface,"进行 OPG")
and send(esinterface,"进行参考打印")
```

note: 切断 3 号烧嘴

习 题 三

1. 简述命题与谓词, 命题逻辑与谓词逻辑之异同。
2. 对下面的公式分别给出使其满足的及不可满足的一组解释:
 - (1) $\forall x \exists y \forall z (S(x, c) \rightarrow T(x, y, z))$
 - (2) $\exists y \forall x \forall z (S(x, c) \rightarrow T(x, y, z))$
 - (3) $\forall x \forall y (S(x, y) \rightarrow S(y, x))$

3. 转化下列谓词公式为子句及 HORN 子句形式。

- (1) $\forall x \forall y (P(x, y) \rightarrow R(x, y))$
- (2) $\forall x \forall y P(x, y) \wedge (Q(x, y) \rightarrow R(x, y))$
- (3) $\forall x \exists u \forall y \forall z R(x, y, z)$

4. 用归结原理证明下式中 G 是 $F1, F2, F3$ 的逻辑结论：

$$F1: \forall x E(x) \wedge \sim \forall (x) \rightarrow \exists y [S(x, y) \wedge C(y)]$$

$$F2: \exists x P(x) \wedge E(x) \rightarrow \forall y [S(x, y) \rightarrow P(y)]$$

$$F3: \forall x (P(x) \rightarrow \sim \forall (x))$$

$$G: \exists x (P(x) \wedge C(x))$$

5. 已知每条鲨鱼都吃人；所有大白鱼都是鲨鱼；有些大白鱼生活在深水里；任何一个被深水鱼吃了的人都是悲惨的。试用谓词逻辑表示此问题，并用归结原理证明有些人是悲惨的。

6. 用基于规则的正向演绎推理方法证明：两个全等三角形的各对应角相等。

7. 用基于规则的反向演绎推理方法证明：各对应边相等的三角形是全等三角形。

8. 用基于系统状态的时序逻辑表达你所见到的一些与时间有关的物理现象。

9. 举例简述单调逻辑与非单调逻辑之异同。

第四章 产生式系统

第一节 产生式系统概述

产生式系统(Production System)的概念由逻辑学家 Post 1943 年提出,用于计算形式体系中的符号串替换运算。Post 认为,任何数学或逻辑系统,都可简化为一系列规则,在规则中指定如何把一个符号串变成另一个符号串,即给定一个输入符号串通过产生式规则可产生一个新的串,而不管其串的物理意义。Post 的产生式系统对于符号变换是有用的,但由于缺乏控制策略,不适合开发实际的应用系统。1954 年,Markov 在字符串替换的研究中,对 Post 的产生式系统作了改进,提出了产生式系统的控制策略,根据规则的优先级来确定其执行的顺序。1957 年 Chomsky 在自然语言结构的研究中提出了文法分层概念,利用一系列产生式规则来描述每层文法的语言生成规则,即重写规则。1972 年,Newell 和 Simon 重新提出了产生式系统,作为人类心理活动中信息加工过程研究的基础,并用它来建立人类问题求解行为的模型。现在,产生式系统已经成为人工智能系统中最典型的一种基本结构,是专家系统及其他应用人工智能系统中最自然的知识表示及推理的基本模型。

一个典型的产生式系统由规则库、工作存储器、控制器三大部分组成。

1. 规则库(知识库)

规则库包含着一系列产生式规则,规则形如:

前件 → 后件

规则的左边称为前件(LHS),表示规则成立时必须满足的条件;规则的右边叫做后件(RHS)表示规则成立时导出的结论或要采取的动作。前件和后件可以用命题或谓词来表示。符号“→”表示变换或者推导。例如,规则 $P \rightarrow Q$ 意为若前件 P 满足,则可运用该规则推导出结论 Q,并把它放入已证明的结论集,或执行 P 所代表的动作。运用一组产生式规则协同工作,把一个产生式推导出的结论作为另一个产生式的前提使用,重复进行推理过程,就可完成定理证明或其他问题的求解。

现在,产生式规则已演变为形式:

IF 条件 THEN 动作或结论

例 1: IF 天阴 and 空气中湿度很大 THEN 可能要下雨

例 2: IF 一种可燃性气体溢出了 THEN 报告消防队

2. 工作存储器

工作存储器又称为动态数据库、综合数据库、短期数据库缓冲器,用它来存储所求解问题的初始状态及已知事实,推理的中间结果以及结论。随着产生式系统问题求解(推理)过程的进展,工作存储器的有些内容(如推理的中间结果)动态变化。工作存储器是产生式系统中主要的数据结构,可以通过简单的表、数组、带索引的文件结构、关系数据库等来实现。

3. 控制器

控制器又称为规则解释器,它控制系统的运行和推理过程,包括:规则扫描的起点和顺序安排;规则前件与工作存储器中事实的模式匹配;工作存储器的状态更新;多条规则被触发时的冲突消解;推理终止条件的判定等。

第二节 产生式系统的工作周期

通常,一个产生式系统的工作周期由模式匹配、选择、执行三个阶段组成,如图 4-1 所示。

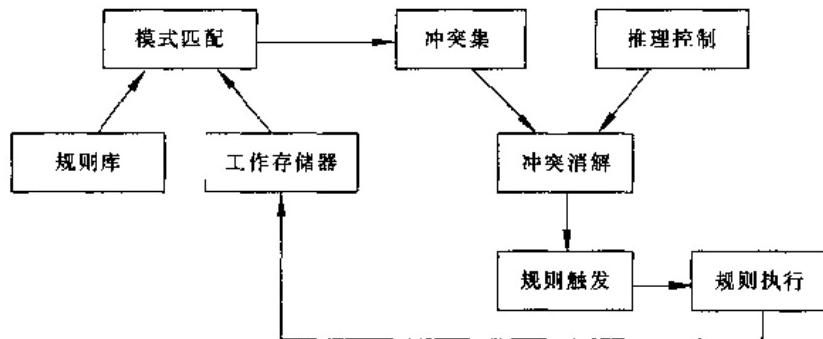


图 4-1 产生式系统的工作周期示意图

工作周期中各阶段的主要工作简述如下:

1. 匹配(合一)。由顶向下(即从知识库中第一条规则开始)依次扫描规则库中所有规则,逐一比较工作存储器的所有元素与所有规则的前件,以搜索满足条件的规则。若一条规则前件中的所有条件都与工作存储器的当前事实匹配成功,则把此规则放进冲突集中,然后进行下一条规则的检测,直到规则库中所有规则都被检测。
2. 选择(冲突消解)。多条规则同时被匹配的情况称为冲突,这时,要根据预先确定的评价准则,求出所冲突规则的优先度,决定选择(触发)那一条规则。

有多种冲突消解的策略,如:

- 按匹配成功的次序选择:优先选择最先匹配成功的规则。
- 按优先权选择:优先选择优先权最高的规则。
- 按详细程度选择:优先选择前提部分描述最详细的规则。
- 按执行的次序选择:优先选择最近执行的规则。
- 按新事实选择:优先选择与最新加到工作存储器中的事实有关的规则。
- 按规则是否使用过选择:优先选择原先没有用过的规则。

以上策略各有利弊,实际系统常常组合使用不同的方法。

3. 执行(动作)。把所选择规则的结论添加到工作存储器,作为新的事实。

运行时,推理机制重复这三个阶段的循环,根据规则库中的知识及工作存储器的事实,不断由已知的前提推出未知的结论,并记录到工作存储器中,作为新的前提或事实继续推理过程,直到推出最终结论。

例:某树类型辨识产生式系统,可模拟植物学家的思维过程,在一系列产生式规则的指导下,通过某些线索(如叶子的形状)来推断树的类型。该系统由规则库、工作存储器、控制器三部分组成。控制器仅提供了两个函数:`on_vn(x)`用来测试树的识别特征 `x` 是否存在于工作存储器 VM 中;`put_on_wm(x)`在一条规则被触发后,把树的识别特征 `x` 添加到 VM 中。规则库

包含的基本规则如下：

- R1: if on_vm(叶子脱落)then put_on_wm(落叶树)
- R2: if on_vm(叶子保持)then put_on_wm(常青树)
- R3: if on_vm(阔叶 and 非银杏)then put_on_wm(被子植物)
- R4: if on_vm(针叶)then put_on_wm(裸子植物)
- R5: if on_vm(一子叶)then put_on_wm(单子叶植物)
- R6: if on_vm(二子叶)then put_on_wm(双子叶植物)
- R7: if on_vm(单子叶植物)or on_vm(双子叶植物)then put_on_wm(被子植物)
- R8: if on_vm(松树球果)then put_on_wm(裸子植物)
- R9: if on_vm(二针叶)or on_vm(三针叶)or on_vm(五针叶)or on_vm(簇针叶)
 then put_on_wm(针叶)
- R10: if on_vm(被子植物)and on_vm(落叶树)and on_vm(叶子密集)
 then put_on_wm(糖槭)
- R11: if on_vm(被子植物)and on_vm(常青树)and on_vm(叶子密集)
 then put_on_wm(冬青树)
- R12: if on_vm(被子植物)and on_vm(落叶树)and on_vm(复合叶子)
 then put_on_wm(核桃树)
- R13: if on_vm(裸子植物)and on_vm(常青树)and on_vm(三针叶)
 then put_on_wm(Ponderosa 松树)
- R14: if on_vm(裸子植物)and on_vm(落叶树)and on_vm(簇针叶)
 then put_on_wm(落叶松树)
- R14: if on_vm(裸子植物)and on_vm(落叶树)and on_vm(簇针叶)
 then put_on_wm(落叶松树)
- R15: if on_vm(裸子植物)and on_vm(常青树)and on_vm(五叶叶)
 then put_on_wm(白松树)

控制器以适当的顺序应用以上规则,模拟植物学家的思维过程,按照产生式系统的工作周期运行,进行树类型的辩识工作。假设系统启动后,通过观察已经获得了三个事实:松树球果、簇针叶、落叶树,并把它们添加到了 WM,使 VM 包含三个元素,即:

$$VM = (\text{松树球果}, \text{簇针叶}, \text{叶子脱落})$$

在产生式系统工作的第一周期中,模式匹配阶段把所有规则的前件与 VM 中的三个元素进行比较测试,发现规则 R1、R8、R9 匹配成功,把这三条规则放入冲突集中。在冲突消解阶段,根据本系统的冲突消解策略——优先触发编号最小,且其后件部分不在 VM 中的规则,触发了规则 R1。在执行阶段执行 R1 后件定义的动作,其结论“落叶树”被添加到 VM 中。于是,在第一周期结束时,VM 中包含四个元素,即:

$$VM = (\text{落叶树}, \text{松树球果}, \text{簇针叶}, \text{叶子脱落})。$$

在第二周期中,重新检查所有规则,进行模式匹配,仍然是规则 R1、R8、R9 匹配成功,根据冲突消解策略,虽然 R1 编号最小,但因其结论“落叶树”已存在于 VM 中,不能被触发,于是,编号次小的规则 R8 触发,其结论“裸子植物”添加到了 VM 中。这样,第二周期结束时,VM 中包含五个元素,即:

$$VM = (\text{裸子植物}, \text{落叶树}, \text{松树球果}, \text{簇针叶}, \text{叶子脱落})。$$

在第三周期中,规则 R1、R8、R9、R14 匹配成功,根据冲突消解策略,规则 R9 被触发,VM 中添加了其结论“针叶”。于是,在第三周期结束时,VM 中包含了六个元素,即:

$$VM = (\text{针叶}, \text{裸子植物}, \text{落叶树}, \text{松树球果}, \text{簇针叶}, \text{叶子脱落})。$$

在第四周期中,规则 R1、R4、R8、R9、R14 匹配成功,规则 R14 被触发,VM 中添加了其结论“落叶松树”。于是,第三周期结束时,VM 中包含七个元素,即:

$$VM = (\text{落叶松树}, \text{针叶}, \text{裸子植物}, \text{落叶树}, \text{松树球果}, \text{簇针叶}, \text{叶子脱落})。$$

在第五周期中,仍是规则 R1、R4、R8、R9、R14 匹配成功,但这些规则的结论均已包含在 VM 中,在冲突集中找不到可以应用的规则,于是产生式系统的推理过程结束,获得了系统的最终结论:由初始事实所推断出的树是美丽的落叶松树。

第三节 产生式系统的控制策略

基本产生式系统的控制策略主要有:

- 不可撤回策略:在搜索过程中,利用所求解问题给定的某些局部知识选择可应用的规则,进行匹配、冲突消解、执行等操作,接着根据工作存储器的新状态选择另一条规则进行有关操作,搜索过程一直往前进行,而不允许撤回已经选择的规则。

- 回溯策略:在从冲突集选择一条规则执行时,建立其回溯点,保留其搜索路径,若以后发现此规则的执行会阻扰或延迟到达推理目标,则放弃此规则,沿原路径返回,重新选择另一条规则,继续搜索过程。

- 图搜索策略:用图来表示问题求解过程,图中结点代表问题的状态,结点间的弧代表所应用的规则。用某种方法选择应用规则,并以图结构记录状态变化过程,直到求出问题的解答。其搜索过程可看作从初始问题图中求出含有解路径的子图。

产生式系统的问题求解过程事实上就是对解空间的搜索过程,又称为推理过程。根据推理过程进行的方向,或说是搜索的方向可分为正向推理、反向推理、正反向混合推理。

一、正向推理

正向推理又称为数据驱动推理,前向链接推理,其推理基础是逻辑演绎的推理链,它从一组表示事实的谓词或命题出发,使用一组推理规则,来证明目标谓词公式或命题是否成立。例如,有规则集如下:

$$\text{规则 1: } P_1 \rightarrow P_2$$

$$\text{规则 2: } P_2 \rightarrow P_3$$

$$\text{规则 3: } P_3 \rightarrow q_3$$

规则中的 P_1, P_2, P_3, q_3 可以是谓词公式或命题。设在工作存储器中已有事实 P_1 ,则应用这三条规则进行正向推理,即从 P_1 出发推导出 q_3 的过程如图 4-2 所示。

实现正向推理的一般策略是:先提供一批数据(事实)到工作存储器中,系统利用这些事实与规则的前提匹配,触发匹配成功的规则,把其结论作为新的事实添加到工作存储器中。继续上述过程,用更新过的工作存储器的所有事实再与规则库中另一条规则匹配,用其结论再次修改工作存储器的内容,直到没有可匹配的新规则,不再有新的事实加到工作存储器为止。

前面已经指出,前件和后件可以用命题或谓词来表示,当它们是谓词时,全局规则前提与

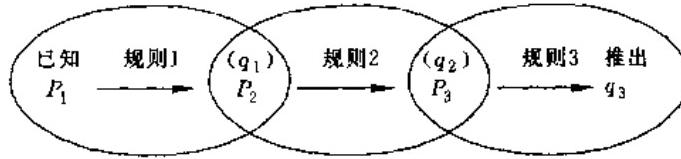


图 4-2 正向推理链接过程示意图

工作存储器中的事实匹配成功是指：对前件谓词中出现的变量进行某种统一的置换，使置换后的前件谓词成为工作存储器中某个谓词的实例，即实例化后前件谓词与工作存储器中某个事实相同。执行后件是指：当前件匹配成功时，用前件匹配时使用的相同变量，按同一方式对后件谓词进行置换，并把置换结果（后件谓词实例）加进工作存储器。实现产生式系统正向推理的一种算法示意如下：

Algorithms Forward_Inference; etp

(1) 读入初始数据(事实)集到工作存储器；

(2) Repeat

{

 取出第 i 条规则；

 规则 i 所有前件与工作存储器的所有事实进行比较；

 if 规则匹配成功 then 把规则加入冲突集；

 if 冲突集为空 then goto 3；

 冲突消解；

 把所选择规则的结论加入工作存储器；

 if 到达目标结点 then goto 3；

 i=i+1；

}

(3) 结束。

正向推理举例：设某产生式系统用于用户购买车辆的咨询，其规则库的内容如下：

R1: if family_size=large
and budget= moderate
and workload=normal
then car=minivan

R2: if family_size=large
and workload=high
and budget= moderate
or budget=high
then car=van

R3: if family_size=small
or family_size=medium
and budget=low
or budget=moderate
and workload=normal

```

        then car=sedan
R4: if      family_size=small
      and    heavy_pulling=yes
      then   car=pickup
R5: if      mileage=high
      or     heavy_pulling=yes
      then   workload=high
R6: if      mileage=moderate
      and    family_size=large
      then   workload=high
R7: if      heavy_pulling=no
      and    mileage!=high
      and    family_size=small
      or     family_size=medium
      then   workload=normal
R8: if      family_member<(4)
      then   family_size=small
R9: if      family_member<(6)
      then   family_size=medium
R10: if     family_member>=(6)
       then  family_size=large

```

工作存储器的初始内容为：

$VM = \{family_member = (6), budget = moderate, mileage = moderate, move_often = yes\}$ ，
正向推理过程如下：

推理机按规则出现的次序扫描规则库，寻找与初始事实相匹配的规则，首先检测规则 R1。R1 第一个条件子句是“family_size=large”，但在 VM 中找不到属性 family_size 的值，无法确定此条件子句的值，以致不能确定 R1 是否为真，于是推理机把 R1 放进待测试规则表，转向下一条规则—R2 的测试。由于 VM 中也没有关于 R2 第一个条件子句属性“family_size”的值，无法确定 R2 是否为真，R2 也被放进待测试规则表，推理机依次转到对规则 R3、R4 的测试。经检查 VM，R3、R4 的取值也无法确定，于是推理机依次把 R3、R4 放进待测试规则表，转而测试规则 R5。VM 中存在的事实“mileage=moderate”与 R5 第一个条件子句“milceage=high”的内容不匹配，导致该条件子句失败。因 R5 的两个条件是“或”关系，只要其中之一为真，R5 即可为真，于是推理机接着测试 R5 第二个条件属性 heavy_pulling 的值。但 VM 中没有关于 heavy_pulling 的事实，R5 的值无法确定，被放进待测试规则表，推理机转而测试规则 R6。因其第一个条件为真，第二个条件无法确定，R6 被放进待测试规则表。接着规则 R7 也被放进待测试规则表，推理机又转向规则 R8、R9。经测试，规则 R8、R9 失败，推理机转向对 R10 的测试。VM 中的事实“family_member=(6)”与 R10 的条件子句匹配成功，R10 被触发，其结论“family_size=large”添加到了 VM 中。

到此为止，推理机已对规则库中的所有规则进行了一遍检查测试，其中两条规则(R8、R9)

失败,一条规则(R10)触发,七条规则(R1、R2、R3、R4、R5、R6、R7)被放进待测试规则表,VM内容被修改为:

```
VM = (family_member = (6), budget = moderate, mileage = moderate,
move_often = yes, family_size = large)
```

推理机进入第二个工作周期后,取出待测试规则表中的七条规则,对它们重新进行测试。测试结果是R1、R2、R5、R7的值仍无法确定,被放进待测试规则表;R3、R4失败;R6与VM中的事实匹配成功,被触发,其结论“workload=high”被添加到VM。

推理机第二个工作周期结束时,待测试规则表中还有四条规则(R1、R2、R5、R7),VM内容被再次修改为:

```
VM = (family_member = (6), budget = moderate, mileage = moderate,
move_often = yes, family_size = large, workload = high)
```

接着推理机开始第三个周期循环,取出待测试规则表中的四条规则,对它们再次进行测试。测试结果是R1失败;R2与VM中的事实匹配成功,被触发,其结论“car=van”被添加到VM;R5、R7的值仍无法确定,被放进待测试规则表;VM内容成为:

```
VM = (family_member = (6), budget = moderate, mileage = moderate,
move_often = yes, family_size = large, workload = high, car = van)
```

在第四个周期中,推理机试图对当前测试规则表中的二条规则(R5、R7)再次进行测试。由于这两条规则的条件中都包含有属性“heavy_pulling”,而现在VM中仍然没有关于“heavy_pulling”的事实,无法对它们进行测试,R5、R7的值仍无法确定,只好再放进待测试规则表。在此周期中VM内容没有变化,系统找不到可以应用的规则,于是推理机停止推理过程,系统给出推理结果(R10的结论):“car=van”,即根据用户的情况,应该购买运货车(van)。

二、反向推理

反向推理又称为目标驱动推理,后向链推理,其基本原理是从表示目标的谓词或命题出发,使用一组规则证明事实谓词或命题成立,即提出一批假设(目标),然后逐一验证这些假设。例如,仍用本节一、例中的三条规则,应用反向推理方法,从 P_1 出发推导出 q_3 的过程如图4-3所示:

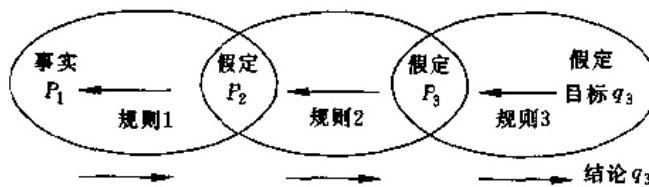


图4-3 反向推理链接过程示意图

首先假定目标 q_3 成立,由规则3($P_3 \rightarrow q_3$),为证明 q_3 成立,须先验证 P_3 是否成立;但工作存储器中没有事实 P_3 ,所以假定子目标 P_3 成立;由规则2($P_2 \rightarrow P_3$),应验证 P_2 ;同样,由于数据库中没有事实 P_2 ,假定子目标 P_2 成立;由规则1($P_1 \rightarrow P_2$),为验证 P_2 成立,须先验证 P_1 。因为工作存储器中有事实 P_1 ,所以假定的目标 P_2 成立,因而 P_3 成立,最终导出结论 q_3 确实成立。

反向推理的具体实现策略是:先假定一个可能的目标,系统试图证明它,看此假设目标是否在工作存储器中,若在,则假设成立。否则,看这些假设是否证据(叶子)结点,若是,向用户询

问,若不是,则再假定另一个目标,即找出结论部分中包含此假设的那些规则,把它们的前提作为新的假设,并试图证明它。这样周而复始,直到所有目标被证明,或所有路径被测试。用产生式系统实现反向推理的一种算法示意如下:

Algorithm Backward_Inference;

1. 读入初始目标 g;
 2. if g 在工作存储器中
 3. then goto 10
 4. else if g 是叶子结点
 5. then {向用户询问 g;}
 6. goto 2; }
 7. else goto 10;
 8. 取出结论部分包含 g 的规则集 S;
 9. while S 不为空 do

取 S 中一条规则的前件 P 作为新的子目标。

用工作存储器中的事实对 P 进行匹配。

if P 为真 then

{

把 R 的结论部分添加到工作存储器：

从 S 中去掉 R;

2

9. goto 2;

10. 结束。

反向推理举例：某产生式系统用于桌面出版系统配置咨询，其规则库内容如下：

R1: if budget_considerations = ok

and hardware = found

then find-rec = ok

R2: if budget_ceiling = high

or budget >1000

then budget_considerations = ok

if hardware != fc

then find_rec != ok

if find_rec != ok

then advice = "There's a problem"

if budget_considerations != ok

then advice = "Can't

if printer = known

and monitor =known

and computer = known

假定系统给定的初始目标是 find-rec, 工作存储器(WM)的初始内容为

VM=(printer=known, monitor=known, computer=known)

则系统进行反向推理的过程如下：

(1) 由于反向推理由目标驱动, 仅考虑直接与目标有关的规则和事实, 因此推理机首先搜索规则库, 寻找包含初始目标“find-rec”的规则, R1符合条件, 被选中。

(2) 因为 R1 包含有两个条件, 推理机先检查其第一个条件, 在 VM 中搜索属性 budget_considerations 的值, 但 VM 中没有关于 budget_considerations 的事实, 由于此条件不是叶子结点, 于是把 budget_considerations 作为当前目标, 以便搜索可能包含 budget_considerations 值的其他资源, 而初始目标 find-rec 被压入暂存目标堆栈中(当对 budget_considerations 的搜索终止时, 可将初始目标 find-rec 从暂存目标堆栈中弹出, 恢复为当前目标)。

(3) 搜索规则库, 寻找包含目标“budget_considerations”的规则, R2被选中。

(4) 检查 R2的第一个条件, 测试其属性 budget_ceiling 的值是否为“high”。推理机搜索 VM, 试图确定 budget_ceiling 的值, 但 VM 中没有关于 budget_ceiling 的信息, 于是把 budget_ceiling 作为当前目标, 而把 budget_considerations 压入暂存目标堆栈中。

(5) 搜索规则库, 寻找包含目标“budget_ceiling”的规则, 没有找到。

(6) 向用户询问“budget_ceiling”的值, 用户回答“unknown”, R2的第一个条件子句匹配失败。于是 budget_ceiling 不再作为当前目标, 被从当前目标存储器中移去, 而把 budget_considerations 从暂存目标堆栈中弹出, 恢复为当前目标。

(7) 由于 R2的两个条件是“或”关系, 只要其中之一为真, R2即可为真, 所以推理机开始考虑 R2的第二个条件, 测试其属性 budget 的值, 把 budget 作为当前目标, budget_considerations 再次被压入暂存目标堆栈中。但在 WM 中找不到关于“budget”的事实, 于是询问用户。

(8) 用户回答“2000”, 此值被赋给属性 budget, 并把“budget = 2000”作为新事实添加到 VM。由于 budget 已有值约束, 不再作为当前目标, budget_considerations 再次从暂存目标堆栈中弹出, 恢复为当前目标。

因为 VM 发生了变化, 推理机重新测试 R2的第二个条件, 现在“budget>1000”的条件成立, R2为真, 被触发, 其后件属性“budget_considerations”被赋值为“ok”, 并添加到 WM 中。

(9) 由于 budget_considerations 已获解, 不再作为当前目标, 初始目标 find_rec 从暂存目标堆栈中弹出, 恢复为当前目标, 推理机返回到 R1, 测试 R1第一个条件的值, 测试结果为真(因为 R2为真, budget_considerations=ok 为真)。

(10) 考虑 R1的第二个条件, 测试属性 hardware 的值, 在 WM 中没有发现关于“hardware”的事实, 于是取 hardware 为当前目标, 初始目标 find_rec 再次被压入暂存目标堆栈。

(11) 搜索规则库, 寻找包含目标“hardware”的规则, R6被选中。

(12) 测试 R6的三个条件。

(13) R6的三个条件与 WM 中的三个事实匹配成功, R6为真, 被触发, 其结论“hardware=found”被添加到 WM 中。

(14) 从暂存目标堆栈中弹出初始目标 find_rec, 恢复其为当前目标, 测试 R1第二个条件的值, 因为 VM 中已有了“hardware=found”的事实, 此条件为真。

(15) R1的二个条件均为真, R1为真, find-rec 被赋值“ok”。

由于系统初始目标 find-rec 已经确定, 所有当前目标已被释放, 暂存目标堆栈中已没有其他目标, 于是推理机停止运行, 系统给出推理结论: “find-rec=ok”, 即桌面出版系统配置已经

成功。

通过前面的介绍可知,正、反向推理的各有其特点和适用场合。正向推理由数据驱动,从一组事实出发推导结论,其优点是算法简单、容易实现,允许用户一开始就把有关的事实数据存入数据库,在执行过程中系统能很快取得这些数据,而不必等到系统需要数据时才向用户询问。其主要缺点是盲目搜索,可能会求解许多与总目标无关的子目标,每当工作存储器内容更新后都要遍历整个规则库,推理效率较低。因此,正向推理策略主要用于已知初始数据,而无法提供推理目标,或解空间很大的一类问题,如监控、预测、规划、设计等问题的求解。

反向推理由目标驱动,从一组假设出发验证结论,其优点是搜索目的性强,推理效率高。缺点是目标的选择具有盲目性,可能会求解许多为假的目标,当可能的结论数目很多,即目标空间很大时,推理效率不高;当规则的后件是执行某种动作(如打开阀门、提高控制电压等)而不是结论时,反向推理不便使用。因此反向推理主要用于结论单一或于已知目标结论,而要求证实的系统,如选择、分类、故障诊断等问题的求解。下面给出正反向推理策略的简要比较表:

反向推理	正向推理
目标驱动	数据驱动
从可能的解答出发,向后推理验证解答	从一组数据出发向前推导结论
由询问关于目标状态的一个问题而启动	从一个事件启动
可解释其推理过程	不能解释其推理过程
由顶向下推理	由底向上推理
典型系统:PROLOG	典型系统:CLIPS, OPS,

三、正反向混合推理

正反向混合推理是为了克服正向推理与反向推理各自的缺点,综合利用其优点而提出的。有多种混合推理策略,其中的一种是通过数据驱动帮助选择某个目标,即从初始证据(事实)出发进行正向推理,而以目标驱动求解该目标,通过交替使用正反向混合推理对问题进行求解。其控制策略比前两种方法都要复杂。美国斯坦福研究院 AI 中心研制的基于规则的专家系统工具 KAS 就是采用正反向混合推理的产生式系统的一个典型例子。下面给出用产生式系统实现正反向混合推理的一种典型示意算法:

```
Algorithm Hybrid_Inference;  
Repeat  
  (1) 读入初始事实到工作存储器;  
  (2) 从已知事实出发进行正向推理,导出部分结果;  
  (3) 选出一个目标 G;  
  (4) 从 G 出发进行反向推理,验证 G 是否成立;  
Until 求出问题的全部解答。
```

第四节 典型的产生式系统CLIPS

CLIPS 是美国航天局(NASA)所属约翰逊空间中心人工智能部80年代末用 C 语言开发的。
• 68 •

通用专家系统工具,是典型的高效正向推理的产生式系统,可在从 PC 到 GRAY 巨型机的各种计算机上运行。

一、CLIPS 的基本组成与知识表示

CLIPS 的核心由事实库(工作存储器)、规则库、推理机三大部分组成,采用产生式规则作为基本的知识表达模式。

(1) 事实(fact): 用来表示已知的数据或信息。事实是一个 N 元式,由一对圆括号括住的一个或 N 个域组成,这些域的数据可以是三种不同的类型,即:字(以字母打头的字符串)、符号串(括在一对双引号内的一个或多个字符串)、数值(整型数或实型数),域之间用空格分开。所有事实都保存在工作存储器中,所以称事实为工作存储器元素(VME)。

事实举例: (high 100m) 表示“高度100米;
(price is 200 yuan) 表示价格为200元;
(name "Newell") 表示名字是 Newell。

可用命令

(assert 事实名)

把事实添加到事实库(工作存储器)中。CLIPS 为所添加的每个事实分配了一个事实索引(标识符)。也可以使用命令

(deffacts 事实名[注释]事实)

来定义事实,当 CLIPS 系统启动推理时,会把所有用 deffacts 定义的事实自动添加到工作存储器中。

可用命令: (retract(事实索引))

删除指定定事实,用命令 (clear) 删除所有事实。

(2) 规则: 用来表示系统推理的有关知识。CLIPS 中的规则是变形的产生式规则,可用 defrule 命令来定义,其格式如下:

(defrule 规则名[注释]
模式
=>
动作)

以上格式中的事实名和规则名可以是有效的 CLIPS 字;可选项注释是放在一对引号内的任意字符串;模式相当于一般产生式前件(LHS)中的条件式;动作相当于后件(RHS)中的动作或结论;箭头符号“=>意为“then”或“则”。

模式的一般格式与事实类似,也是可包含 N 个域的 N 元式,与事实不同的是在规则模式中可以包含变量。变量的一般表达形式为: ?变量名

变量名用字,即字母打头的字符串表示,如变量 X 可表示为 (?X)。

动作主要用来添加或删除工作存储器中的事实以及控制运行过程。CLIPS 提供的一些动作控制命令包括:

· 变量约束命令 blind: 用于变量的值同表达式的值的约束或连接,可用 blind 建立新变量或修改已在规则中赋值的变量。blind 的格式为:

(blind 变量名 表达式或值)

- 暂停命令 halt,用来暂停规则的执行。
 - 选择命令 if,用来选择动作的执行顺序,命令格式为:
(if 谓词函数式 then 动作序列1 else 动作序列2)
谓词函数式取值为真或假,它可以是 CLIPS 系统预先定义的内部函数,也可以是用户以 C 或其他语言编写,并连接到 CLIPS 的外部函数。CLIPS 系统预先定义的内部谓词函数包括:
 - 逻辑谓词函数:and,or,not
 - 比较谓词函数:eq,neq,=,!=,>=,>,<=,<。其中,eq 及 neq 用于任意值的等于及不等于比较;=及!=仅用于数值的等于和不等于比较。
 - 循环命令 while:用来控制某些动作的重复执行,其格式为:
(while 谓词函数式 [do] 动作序列)
if 及 while 命令的含义与 C 语言相同。
例如,产生式规则

```
if have head-ache (若 头痛)
  and sore throat (嗓子痛)
  and runny nose(流鼻涕)
then take aspirin (则 服用阿斯匹林)
and goto bed (睡觉)
```
- 在 CLIPS 中可定义为:
- ```
(defrule cold "An example rule"
 (have head-ache)
 (sore throat)
 (runny nose)
 =>
 (take aspirin)
 (goto bed))
```

(3)待处理事件表:用于存储匹配成功的(或说是被激活的)规则集合,它相当于一般产生式系统中的冲突集。待处理事件表实际上是一个堆栈,所有激活的规则被按优先级别定义的次序压入堆栈。若新压入规则的优先级小于栈顶规则的优先级,则它被压入到栈的下部,直到所有比它优先级高的规则都在此规则的上面。规则的优先级别由用户在 CLIPS 程序中定义。系统将选择待处理事件表中优先级最高的规则执行。若待处理事件表中没有规则,一般情况下,程序将停止运行。CLIPS 的程序通过 run 命令而启动运行。

## 二、CLIPS 的推理机制

### 1. CLIPS 的推理机制概述

CLIPS 的推理机制是在 OPS(Official Production System)基础上发展起来的,其基本工作原理与 OPS 推理机类似。OPS 是美国卡内基-梅隆大学开发的基于正向推理的产生式系统,自 1975 年推出以来,经过多次修改,影响较大的是 1981 年推出的 OPS5 和 1986 年推出的 OPS83。OPS 和 CLIPS 的基本结构是产生式系统,工作周期有匹配、选择、执行三个主要阶段,其推理机的工作过程为:

- 模式匹配。扫描规则库,把所有规则的模式(LHR)与事实表中的事实进行匹配,检查那些规则的条件能够满足;
  - 激活所有匹配成功的规则,把它们放进待处理事件表中。
  - 弹出堆栈顶部的规则(优先级最高),执行其后件(RHS)部分所规定的动作。
- 重复以上过程,直到待处理事件表变为空。

CLIPS 与一般产生式系统的不同之处在于其推理过程基于下述 RETE 算法。

## 2. RETE 模式匹配算法

RETE 模式匹配算法由美国卡内基-梅隆大学 Charles L. Forgy 1979 年在他关于 OPS 专家系统外壳的博士论文中提出。这个算法的核心就是利用动态匹配图来提高匹配阶段的工作效率,从而提高了产生式系统的运行速度。一些基于规则的语言,如 CLIPS, ART, OPS5, OPS83, FORPS 等均采用了 RETE 算法。

一般产生式系统的模式匹配过程是动态进行的。在每个执行循环中,由于事实库中的事实要被增、删修改,规则模式的真、假性可能要发生变化。为了保证推理的正确性,需要在每个执行周期后,再次搜索工作存储器,对所有规则的真假性再做一次检查。这种策略简单易行,但会耗费大量时间,严重降低系统的运行速度。

研究指出,产生式系统具有时间冗余及模式相似的特性。时间冗余特性指工作存储器中的数据随时间缓慢变化,每个工作周期仅其中少量的事实可能发生改变。模式相似特性指不同的规则以及同一规则的前件之间存在着许多相同的测试。根据这两个特性,RETE 算法把每个产生式系统工作周期的匹配过程都保留下来,而仅对工作存储器中发生的变化重新计算系统状态,以节省推理过程的时间开销。其实现手段是在匹配过程中生成匹配网络图,来保留匹配过程及系统状态,决定下一步的操作。

RETE 算法中的匹配网络图由模式网络和连接网络组成。

模式网络用来进行规则的模式匹配测试,即检查各个模式与工作存储器中的那些事实匹配(每个事实称为一个工作存储器元素,记为 VME)。为了测试实现的方便和高效,所有模式被编译成一个树形结构,每个结点仅有一个输入,根结点是网络的入口。连接到树根的结点表示所有模式的第一个域,连接到第二层的结点表示所有模式的第二个域;……连接到第 n 层的结点表示所有模式的第 n 个域;叶子结点表示模式的终结。每个结点存储一个模式域的测试表达式,用来确定 VME 的一个域是否与规则模式的一个域匹配。换言之,每一个测试条件对应于模式网络的一个域结点,每一模式的所有域结点依次连接起来,构成模式网络的一条匹配链。每条匹配链的终结结点设有一个  $\alpha$  寄存器,记录与此模式匹配的所有 VME。

考虑到某些规则模式的相似特性,为了避免不必要的重复测试检查,在构造模式网络时尽可能地公用树的分支,对前 K 个域相同的两个模式,共享 K 个网络结点。此外,含有相同模式的两条规则,共享一条匹配链和  $\alpha$  寄存器。

连接网络用来比较和检查一条规则 LHS 中各模式间的变量连接,即检查不同模式的同名变量之间的相互约束关系,以保证同名变量取值的一致性。连接网络的每个结点有两个输入,一个是模式网络终结结点的  $\alpha$  寄存器,另一个是记录着与先前的连接相匹配的所有 VME 的  $\beta$  寄存器。构造连接网络时,先比较第一个和第二个模式,得到第一个连接,把结果写进连接结点的  $\beta$  寄存器,然后把此  $\beta$  寄存器的内容与另一个模式相比较,得到第二个连接,把结果写进第二个  $\beta$  寄存器,……直到检查完毕所有模式间的变量连接。

例如,有规则

```

(defrule exa-2
 (match ?y red)
 (data ~green ?y)
 (data ?y ?y)
 (other ?y)
=>)

```

此规则对应的模式网络和连接网络图示于图4-4。

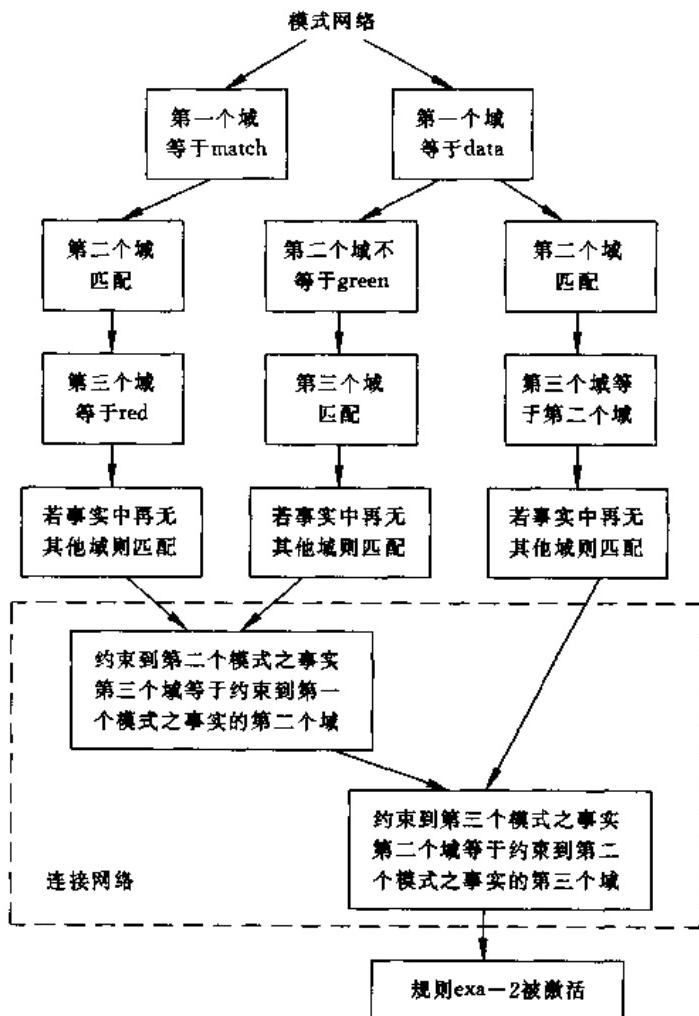


图4-4 规则 exa-2对应的模式网络和连接网络示意图

RETE 网络的匹配过程由在模式网络上的模式匹配和在连接网络上的部分匹配组成。因为模式网络是树形结构,所以用二叉树遍历算法搜索整个网络,实现模式匹配,在搜索过程中进行网络结点和 VME 比较。模式匹配完成后,便开始连接网络上的部分匹配,根据模式匹配过程中各模式  $\alpha$  寄存器内容的变化,检查相应连接网络的结点,并修改其  $\beta$  寄存器的内容。最后,把  $\beta$  寄存器记录的完全匹配成功的规则,送入处理事件表。上述匹配过程是由工作存储器中 VME 的增、删、修改变化所驱动的。

由于 RETE 算法在产生式系统的匹配阶段充分利用了时间冗余及模式相似的特性,保留了匹配过程中获得的大量中间信息,减少了时间开销,提高了推理速度。但是,RETE 算法对产

生式系统速度的提高是有限的。特别是在系统所接受事实相似性较小的情况下,对速度没有明显改善。另外,由于 RETE 算法要保留全部模式匹配状态,需要消耗大量的存储空间。再从时间复杂度来考虑,令  $W$  表示工作存储器元素的总数,  $A$  表示每条规则的前件数,  $R$  表示产生式系统的规则总数,则使用 RETE 算法时,工作存储器元素对时间的复杂度在最好情况下是  $O(1)$ ,在最坏情况下是

$$O(W^{2A-1})$$

规则数对时间的复杂度最好情况下是  $O(\log_2 R)$ ,在最坏情况下是  $O(R)$ 。所以,使用 RETE 算法时,匹配阶段所花费的时间决定于产生式系统的规则数及工作存储器元素数。这样,知识库规模越大,使用 RETE 算法的产生式系统就越慢。

## 第五节 对产生式系统的评价

从前面的讨论可知,产生式系统不仅是一种计算模型和问题求解系统,也是一种实用而强大的知识表示模式,具有以下特点:

- 具有丰富的知识表示能力,可以简单直观的规则方式表达人类的经验性知识。
- 知识表达模块化、结构化,每条规则具有相同的格式且相对独立。规则的组合、修改、增、删比较容易,规则的收集、整理比较方便。
- 不仅可以表达知识,也能表达动作,其结构事实上等价于图灵机。
- 容易排除故障,当系统工作异常时,通过跟踪产生式规则的触发序列,就可容易地发现故障,为系统调试和维护提供便利条件。
- 产生式系统的规则链接推理过程相似于人类求解问题时的逻辑思维过程,因此可把产生式系统用作人类行为的启发式模型,模拟人类的逻辑思维过程。
- 推理方向的可逆性。由于产生式规则的前件和后件结构类似,可同时进行正向和反向推理,即混合推理,当解决复杂问题时,可提供有效的框架结构。
- 控制机构的多样性。由于产生式系统的控制机构对用户是开放的,因此可根据对象领域和欲求解问题的特点设计最佳控制机构。

由于以上优点,产生式系统适合在以下场合应用:

- 知识结构类似于产生式规则的领域,如医疗诊断系统。其特点是领域知识比较零乱,由大量经验性的独立事实和规则组成,缺乏像数学、物理学科那样系统、严格而又简明的理论。
- 领域知识中包含一系列相互独立的动作,因而可以自然地用产生式规则的后件来表达的领域,如医院的病人监护系统。
- 领域知识可方便地从应用中分离出来的领域,如前面介绍的树类型辨识系统,经典分类学等。

某些领域,如数学,具有严格的理论体系,形成一个统一的整体,很难分割开来,所以不适用于用产生式系统表达,但是,并不排除数学中的某些局部知识可以用产生式系统来表达。例如,积分学中的包含了许多彼此独立的积分规则,Slagle 的 SAINT 系统就是利用产生式系统来求解不定积分问题的。

Rychener 指出,若欲求解之问题可视为问题空间中一个状态到另一个状态的变换序列,则可用产生式系统求解。诸如 8 迷(8 Puzzle)、旅行商(traveling salesman)、汉诺塔(Tower of Hanoi)、传教士与食人魔(Missionaries and cannibals)、符号积分(Symbolic integration)、猴子与

香蕉 Monkey and bananas)、三子棋(Tic-tac-toe)等经典人工智能问题的求解,以人类专家知识为基础的专家系统的问题求解,从本质上都可以看作是从初始状态到目标状态的推导变换过程,因而都可用产生式系统来求解。

但是,多年的理论研究与工程实践证明,产生式系统也存在一些重要缺陷,主要是:推理效率低、速度慢、实时性能差,随着规则库规模的增大容易产生组合爆炸等。例如,基于产生式的正向推理系统在已知数据(初始事实)的驱动下,盲目搜索整个规则库,求解了许多与总目标无关的子目标,随着规则库规模的增大,搜索空间急剧加大,可能导致组合爆炸。基于产生式反向推理系统盲目选择系统目标,可能求解了许多为假的总目标,随着规则库规模及搜索空间的加大,导致推理效率急剧下降。正反向混合推理系统虽然在一定程度上限制了搜索范围,提高了推理的灵活性和效率,但仍摆脱不了随着规则库规模的增大容易产生组合爆炸的问题。在每次推理循环中,它们都要进行规则是否匹配的检查,即对所有规则的前件进行检查,然后把所有满足匹配条件的规则放在一起进行冲突消解,选出一条优先级最高的规则执行。由于这个过程包含了大量无效的匹配尝试,浪费了大量系统的推理时间,而且规则数目越大搜索时间越长,由于无效的匹配尝试而浪费的时间也就越多,因而推理效率也就越低。试验表明,产生式系统在匹配阶段所花费的时间大约为产生式系统工作周期的90%,而冲突消解及执行阶段所花费的时间大约占工作周期的10%。O'Reily 及 Cromarty 1985年的分析指出,使用正向链的产生式系统运行时的时间复杂度为指数级,在规则数量较大的情况下,随着推理树深度的增多,将触发组合爆炸。在使用反向链的情况下,推理树深度的每增加一级,将使树的结点数作指数级增加,结点的分支数越大,组合爆炸就发生得越快。因此,以产生式系统为基础而建造的应用智能系统很难适应实时要求。为了提高产生式系统的性能,人工智能研究者作了不懈的努力,试图通过改进匹配算法和并行处理等措施来提高其性能。几种典型的工作如下:

## 一、RETE 算法。

在本章第四节已作了介绍,此处不再重复。

## 二、知识库编译

这种方法1981年由 Neves 及 Anderson 提出。1986年的知识库编译学术会议上给予了肯定。知识库编译是通过优化知识库结构而改进推理系统运行效率的一种方法,它把以显式表示的说明性知识经过编译程序而变换为效率更高的隐含表示的过程性知识。即把规则库中的规则子句翻译为机器代码。可通过查找代码的地址来搜索它们。这样,在匹配阶段进行比较的仅是机器代码,而不是规则子句,因而加快了推理速度。

但是,在知识库编译方面还有许多问题和困难需要进一步解决,如知识库编译的精确定义;源及目标知识结构的形式;深层及浅层知识编译差别;知识库编译的实现技术和工具等。另外,以知识库编译方法为基础的专家系统在运行时所输入的数据也必须变换为相应的机器码,同时知识库的维护也不方便。

### 三、并行处理

通过支持并行处理的硬件及软件来提高产生式系统的效率。并行硬件如流水线处理机,阵列处理机、向量处理机、PROLOG 机、LISP 机、Transputer 等。并行程序设计语言如 Concurrent C,Concurrent PROLOG ,PARLOG 等。以 RETE 算法为基础的并行产生式系统也已出现。Stolfo 等提出了一个高度并行,树状结构的处理器来支持各种人工智能应用,特别是并行产生式系统的应用。他们的原型机称为 DADO 系统。其中 DADO-1 系统由 15 个处理器互联成了一个完全二进树。DADO-2 系统由 1023 个处理器组成,在 DADO 系统上并行实现了 RETE 算法。K.Oflazer 1984 年提出了产生式系统并行处理的划分理论。G.L.Forgy 1986 年提出了规则库系统的并行算法及结构。1987 年 F.Ramnarayan 等发表了产生式系统的并行结构 PESA-1。1991 年 Vishneshwar 等发表了在并行产生式系统中的最小状态空间搜索理论。所有这些研究工作促进了并行产生式系统的发展。但是,并行处理所付出的硬件及软件代价太高,不利于在工业生产过程中,特别是在以通用微型计算机为基础的环境中推广应用。

解决基于产生式的智能应用系统效率低、速度慢、实时性能差问题的另一个卓有成效的途径就是与神经网络技术相结合,利用神经网络的并行处理能力提高系统的性能。这将在本书第十六章介绍。

## 习 题 四

1. 简述产生式系统的基本结构和特点,试提出用某种高级语言(例如 C)实现工作存储器及规则库的方案(包括数据结构和基本算法)。
2. 为什么可用产生式系统来建立人类问题求解行为的模型?如何建立这种模型?
3. 举例说明基于正向推理以及反向推理的二个产生式系统的例子,指出其适用场合。
4. 简述 RETE 算法是如何改进产生式系统性能的?其局限性是什么?试举例说明改进产生式系统性能的其他方法。
5. 简述正反向混合推理系统的工作原理,指出其实现的难点。

## 第五章 基于结构化表示的问题求解

知识的结构化表示方式的特征是结构与层次清楚。其主要优点是表示的知识自然、直观，有利于提高问题求解的效率。框架系统和语义网络是人工智能中最常用的两种结构化知识表示方法，而面向对象的表示方法是很有发展前途的结构化知识表示方法。

### 第一节 语义网络

语义网络是 J. R. Quillian 1968 年在博士论文中作为人类联想记忆的一个显式心理学模型最先提出的。他主张在处理自然语言词义理解问题时，应当把语义放在第一位，一个词的含义只有根据它所处的上下文环境才能准确地把握。基于 Quillian 的工作，西蒙于 1970 年正式提出了语义网络这个概念。自七十年代中期以来，语义网络已在专家系统、自然语言理解等领域得到了广泛的应用。

#### 一、语义网络的知识表示

##### (一) 基本命题的语义网络表示

语义网络形式上是一个有向图，由一个结点和若干条弧线构成，结点和弧都可以有标号。结点表示一个问题领域中的物体、概念、事件、动作或状态，弧表示结点间的语义联系。

在语义网络知识表示中，结点一般划分为实例结点和类结点（概念结点）两种类型。如“汽车”这样的结点是类结点，而“我的汽车”则是实例结点。有向弧用于刻画结点之间的语义联系，是语义网络组织知识的关键。由于语义联系非常丰富，不同应用系统所需的语义联系和种类及其解释不尽相同。比较典型的语义联系有：

###### 1. 以个体为中心组织知识的语义联系

以个体为中心来组织知识，其结点一般都是名词性个体或概念，通过实例、泛化、聚集、属性等联系作为有向弧来描述有关结点概念之间的语义联系。

###### (1) 实例联系

用于表示类结点与实例结点之间的联系，通常用 ISA 标识，如图 5-1 所示。



图 5-1 实例联系举例

一个实例结点可以通过 ISA 连接多个类结点，多个实例结点也可以通过 ISA 与一个类结点相连接。

通过类结点表示实例之间的相关性，并使同类实例结点的共同特征通过与此相连的类结点来描述，从而实现了知识的共享，简化了网络的结构。

###### (2) 泛化联系

用于表示类结点（如熊猫）与抽象层次更高的类结点（如哺乳动物）之间的联系，通常用

AKO(A Kind Of)来标识。通过 AKO 可以将不同抽象层次的类结点组织成一个 AKO 层次网络,如图5-2所示。

泛化联系允许低层类结点继承高层类结点的属性,因而一些共同的属性不必在每个低层类结点中重复,从而节省了存储空间。

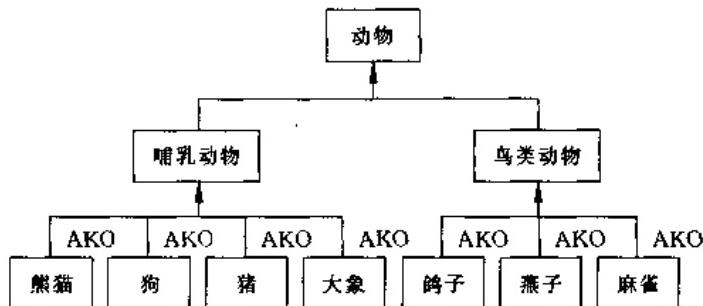


图 5-2 泛化联系举例

### (3) 聚集联系

用于表示与其组成成分之间的联系,通常用 Part-of 表示,如图5-3所示。

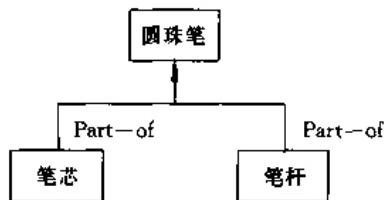


图 5-3 聚集联系举例

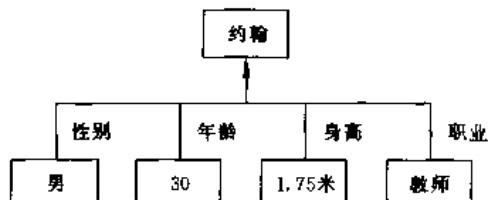


图 5-4 属性联系举例

### (4) 属性联系

用于表示个体、属性及其取值之间的联系。通常用有向弧表示属性,用弧所指向的结点表示属性的值,如图5-4所示。

图5-5是描述桌子的语义网络,其中包含了上述实例、泛化、聚集和属性四种联系。由图可见,以个人为中心来组织知识,其结点一般都是各词性个体或概念,其间的语义联系通过 ISA、AKO、Part-of 以及属性标识的有向弧来实现。

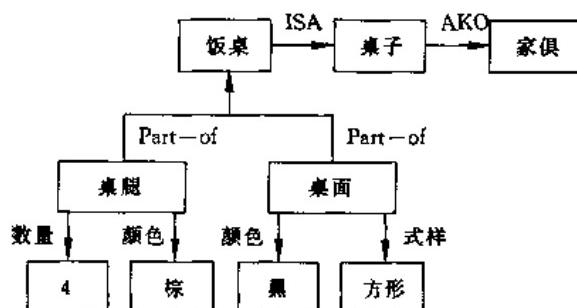


图 5-5 描述桌子的语义网络

## 2. 以谓词或关系为中心组织知识的语义联系

设有  $n$  元谓词或关系  $R(arg1, arg2, \dots, argn)$ , 分别取值为  $a1, a2, \dots, an$ ), 其对应的语义网络可表示为图5-6的形式。

为了表示“有一张木头做的方桌”这一知识可用以个体为中心来组织知识的语义网络来表示,如图5-7(a)所示。图中 Comp 表示材料属性,Form 表示形状属性。也可用以谓词为中心来组织知识的语义网络来表示,如图5-7(b)所示。图中 Comp(桌子,木头)表示“桌子的材料是木头”这一谓词或关系;Form(桌子,方形)表示“桌子的形状是方的”这一谓词或关系。

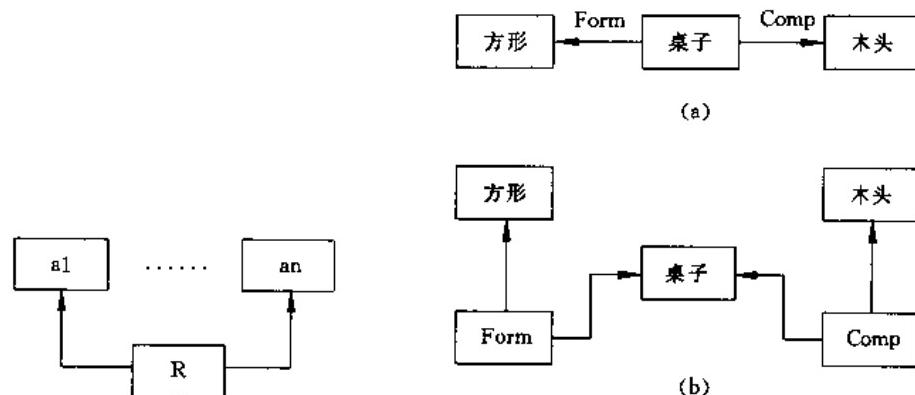


图5-6 关系语义网络表示

图5-7 “有一张木头做的方桌”的语义网络

与个体结点一样,关系结点同样可以划分为类结点和实例结点两种,实例关系结点与类关系结点之间用 ISA 标识。如图5-8所示。图中把动作“give”看作一个三元关系,其属性分别为给予者(giver)、接收者(recipient)和被给的物体(object)。Give1为实例关系结点,give 为类关系结点。

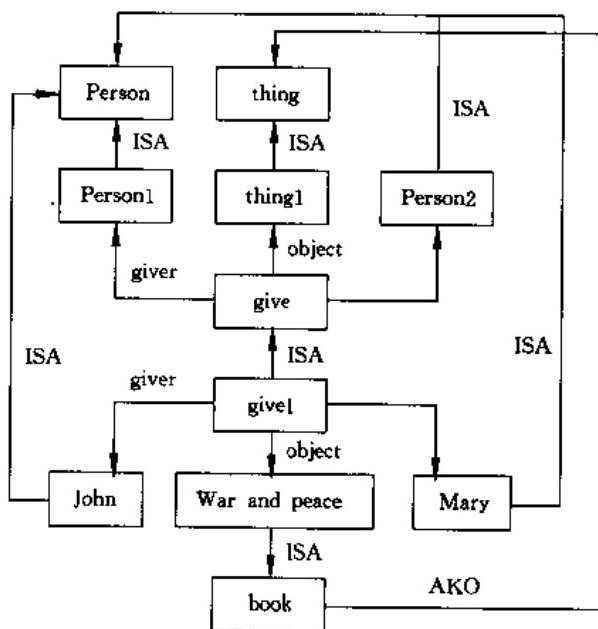


图5-8 关系语义网络举例

## (二) 连接词在语义网络中的表示方法

在稍微复杂的命题中,往往含有“与”、“或”、“非”及“蕴涵”等连接词,因此语义网络还应能表达对应的四种关系。

### 1. 合取

由于语义网络从本质上讲,只能表示二元关系,因此当要用它表示多元关系时,一种可行的办法是把多元关系转化为二元关系的合取。为了进行这种转换,需要引入附加结点,例如用语义网络表示一个三元关系 GIVE(John, Mary, Book),如图5-9所示。

图中结点为“与”结点,与 G 结点相连的 Giver、Recipient 及 Book 三个连接构成合取关系。

## 2. 析取

在语义网络中,如果不加标志,就表示连接之间的关系是合取关系,而对析取、否定和隐含关系,都加有标志。在连接上加注 DIS 析取界限来表示析取关系。例如要表示

$\text{ISA}(A, B) \text{ (Part-of}(B, C)\text{)}$

其语义网络如图5-10所示。如果不加注析取界限,则此网络就可能误解为

$\text{ISA}(A, B) \text{ (Part-of}(B, C)$

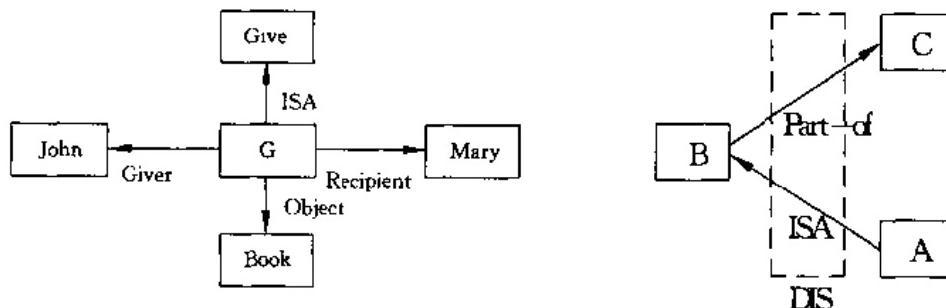


图5-9 合取的语义网络表示

图5-10 析取的语义网络表示

## 3. 否定

在语义网络中否定的表示有两种。对单个关系的否定可在弧的标记前加上否定符号 $\sim$ ,如 $\sim(\text{ISA}(A, B))$ 可表示为图5-11(a)。表示多个关系的合取或析取的否定时,可采用 NEG 界限标注,如图5-11(b)所示。

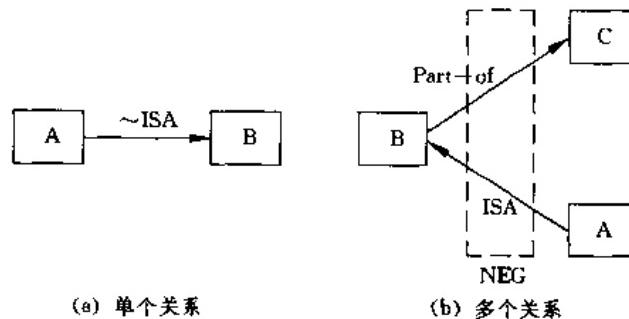


图5-11 否定的语义网络表示

## 4. 蕴涵

在语义网络中,用标注 ANT(antecedent)和 CONSE(consequence)界限分别表示蕴涵前件和蕴涵后件。例如:

Every one who lives at 37 Maple street is a programmer

可用语义网络表示为图5-12。图中在前件部分用 Y 结点表示地址事件,X 结点是一个变量,表示与 Y 事件有关的人;在后件部分建立了一个表示职业事件的结点。一个特定的职业事件是 X 和 Y 的函数,没有必要以新的变量来标识。图中虚线框分别标注出与前、后件有关的弧,并用虚线把两个界限连接起来以表示蕴涵关系。

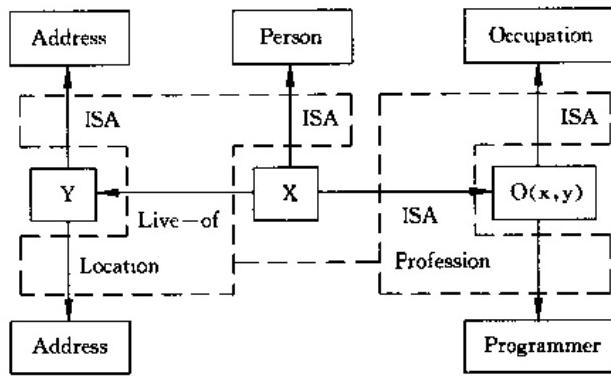


图 5-12 蕴涵的语义网络表示

### (三) 量词在语义网络中的表示

存在量词在语义网络中直接用 ISA 弧表示，而全称量词则需用分块化技术实现。

#### 1. 存在量词的表示

例如要表示

The dog bit the postman

用谓词逻辑可表示为

$$\exists X(DOG(x) \wedge BITE(x, Postman))$$

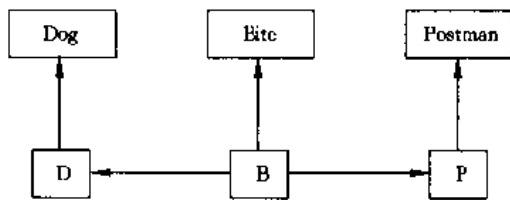


图 5-13 存在量词的语义网络表示

其语义网络表示如图 5-13 所示。图中 D 结点表示一特定的狗，P 表示一特定的邮递员，B 表示一特定的咬人事件（包括攻击者和受害者两部分）。实例结点 D、P 和 B 与相应的概念结点用 ISA 弧线相连。

#### 2. 全称量词的表示

例如下列事实

Every dog has been a postman

用谓词逻辑可表示为

$$\forall X(DOG(x) \rightarrow \exists Y(Postman(Y) \wedge BITE(x, y)))$$

这种含全称量词的表示方法是：把句子中的每一个全称变量进行分割，用一个特定的子空间表示。如 S1U 是一个子空间，它表示一个断言

A dog has a postman

该子空间内不含全称量词，称之为格式（Form）。然后在问题的整个空间中引入一个特殊的弧，称之为  $\forall$  弧。一条  $\forall$  弧表示一个全称量化的变量。这样，上述事实便可表示为图 5-14(a)。其中 GS 表示全称量化的事实概念，GS1 是 GS 的一个实例结点。利用这种分块技术，可进一步实现多个全称量词的逻辑表达。如

$$\forall X(DOG(x) \rightarrow \forall Y(Postman(Y) \wedge (BITE(x, y))))$$

可用图 5-14(b) 表示。

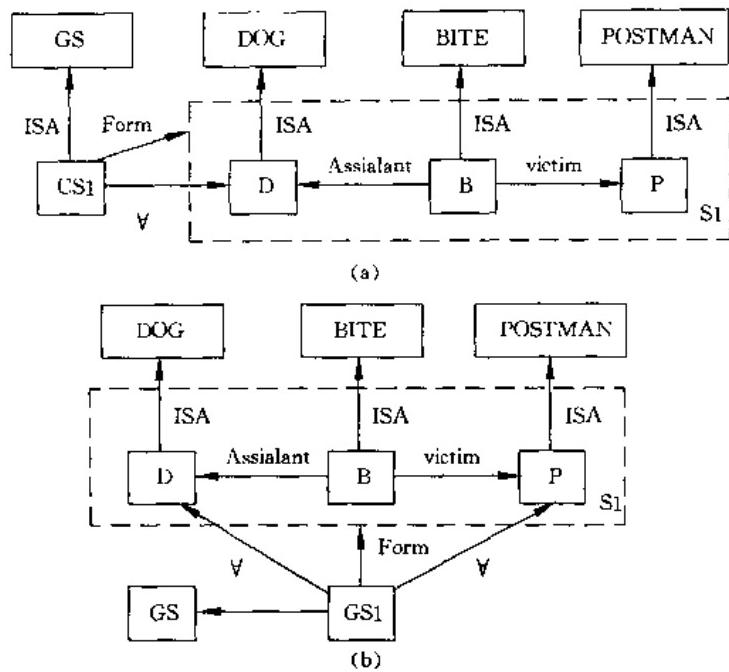


图5-14 全称量词的语义网络表示

## 二、语义网络的推理

目前,大多数语义网络所采用的推理机制主要有两种,即匹配和继承。

### (一) 匹配

在语义网络中,事物是通过语义网络这种结构来描述的,事物的匹配则为结构上的匹配,包括结点和弧的匹配。用匹配的方法进行推理时,首先构造问题的目标网络块,然后在事实网络中寻找匹配。推理从一条弧连接的两个结点的匹配开始,再匹配与该两个结点相连接的所有其他结点,直到问题得到解答。这种方法的例子如图5-15所示,其中(a)为事实网络,(b)为目标网络。

图5-15(b)表示这样一个问题:

What does clyde own?

用(b)这个目标网络去匹配事实网络(a),寻找一个有一条 owner 弧指向 Clyde 的 Own 结点。当找到这样的结点时,owner 弧指向的结点即为上述问题的答案(Nest1),否则,答案便将是:

Clyde doesn't own anything.

当事实网络较大或较复杂时,在匹配算法中可加入一些含有启发式知识的选择器函数,以提供事实网络中哪些结点和弧可以优先考虑匹配和怎样匹配的建议。这种选择器函数能加速匹配的搜索过程。

### (二) 继承

在语义网络中所谓继承是把对事物的描述从概念结点或类结点传送到实例结点,这种推

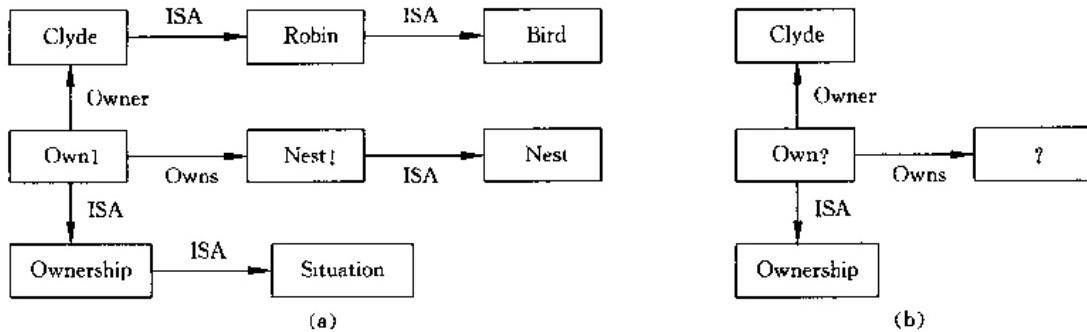


图5-15 网络的匹配过程

理过程类似于人的思维过程,一旦知道了某种事物的身份,便可联想起很多关于这件事物的一般描述。语义网络的继承推理方式有三种:值继承、“默认”继承和“附加过程”继承。

### 1. 值继承

以图5-16为例,来说明怎样使用值继承推理求出 Brick1的形状。作为问题的给定结点 Brick1和弧 Shape,从 Brick1结点出发,检查是否有以其为出发点的 Shape 弧。如果有,则 Shape 指向结点的值即为解;否则依次检查与 Brick1相连的 ISA 弧指向的结点,如果有从这些结点出发的 Shape 弧,则找到解。图中有与 Brick 结点相连的 Shape 弧,它指向结点 Rectangular,即得到 Brick1的形状为 Rectangular 的解。如果从所有通过 ISA 弧与 Brick1相连的这些结点上都没有出发的 Shape 弧,则开始查找 SKO 链指向的结点,看是否有 Shape 弧从那里出发。如此搜索下去,如果直到最后都找不到 Shape 弧,则宣布搜索失败。

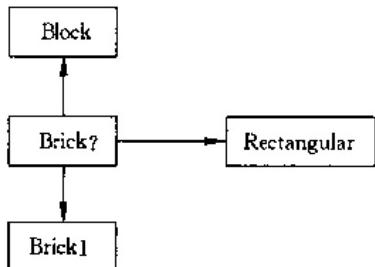


图5-16 语义网络的值继承

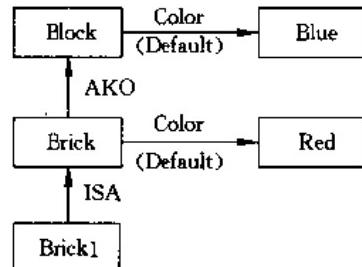


图5-17 语义网络的“默认”继承

### 2. “默认”继承

某些情况下,在对事物所作的假设不是十分有把握时,最好对假设加上“可能”这样的词。例如,头痛可能是感冒,但也不一定是;宝石可能很贵,也不一定是。把这种具有相当程度的真实性但有不能十分肯定的值称为“默认”值。表示该结点的值为“默认”值的方法,是在指向该结点的弧的标注下加上 DEFAULT 标记。图5-17所示网络的含义是,从整体来说,Block 的颜色可能是 Blue,而 Brick 的颜色可能是 Red。

“默认”继承的推理过程类似于值继承,不过搜索的是给定弧标注下带有 Default 标记的弧。

### 3. “附加过程”继承

在某些情况下,对事物的描述不能直接从概念结点或类结点继承而得,但可以利用已知的信息来计算。例如,可以根据体积和密度来计算积木的重量。进行上述计算的程序称为 if-needed 程序,存放在带有 IF-NEEDED 附加标记的弧指向的结点中。例如图5-18(a)所示,一个计算

重量的程序存放在连接 Block 结点的 Weight 弧(下加 IF-NEEDED 标记)指向的结点中。

“附加过程”继承推理的过程也类似于值继承过程。如图5-18(b)所示,为了得到 Brick1 的重量,搜索下加 IF-NEEDED 标记的 Weight 弧,执行 if-needed 程序,然后在 Brick1 结点上添加 Weight 弧,并把 if-needed 程序的执行结果,即重量,填入 Weight 弧指向的结点。

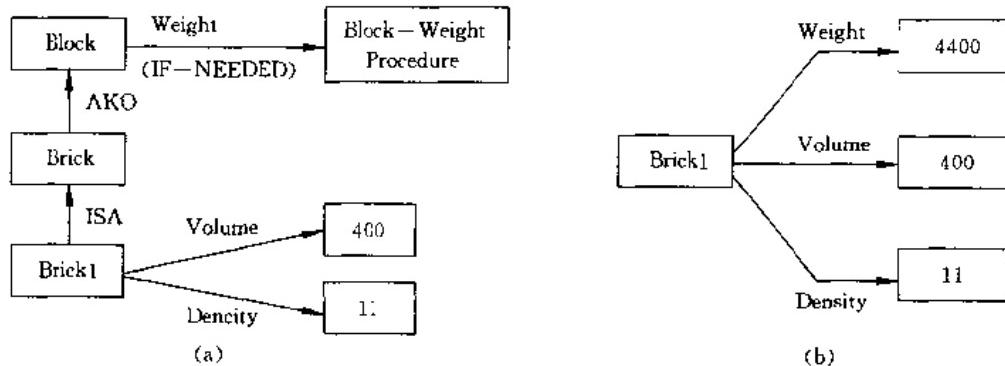


图5-18 语义网络的“附加过程”继承

### 三、语义网络表示的特点与不足

由结点和弧组成的语义网络表达直观、自然,易于理解,其继承推理方式符合人类的思维习惯。语义网络把事物的结构、属性及事物间的联系显式地表达出来,与一个事物相关的事实、特征、关系可以通过相应结点的弧推导出来。基于语义网络表达的系统便于以联想方式实现系统的解释。因此,语义网络是人工智能中一种重要的知识表示方法。

但语义网络表示法试图用结点代表世界上的各种事物,用弧代表事物间的任何联系,其形式过于简单。如果结点间的联系只局限于几种典型的关系,则难以表达较复杂的关系;而增加联系又会大大增加网络的复杂度,相应的知识存储和检索过程就会变得十分复杂。事实上,语义网络的管理和维护也是很复杂的。

此外,语义网络本身没有赋予其结点和弧以确切的含义。推理过程中有时不能区分物体的“类”和“个体”的特点。如一个结点标记为“大学生”,那么它究竟是指“大学生”这个概念,还是所有“大学生”的集合,或是某个特定的大学生呢?这个标记结点可以赋予不同的解释,完全由系统设计者决定。事实上,语义网络结构的语义解释完全取决于推理过程,不同的推理过程对应于不同的解释。由于语义网络结构本身没有语义上的约定,因此它不具备逻辑系统那样的有效性。

## 第二节 框架系统

心理学的研究结果表明,在人类日常的思维和问题求解活动中,当分析和解释新的情况时,常常使用从过去的经验中积累起来的知识。这些知识规模巨大而且以很好的组织形式存储在人类的记忆中。由于过去的经验是由无数个具体事例、事件组成的,人们无法把所有事例、事件的细节都一一存储在大脑中,而只能以一个通用的数据结构的形式来存储。这样的数据结构称为框架。对一个特定的事物,只要把它的特征数据填入框架,该框架就表示了该事物。同时,可以根据以往的经验获得的概念对这些数据进行分析和解释,还可以寻找与该事物有关的统

计信息。

## 一、框架的构成

一个框架(Frame)由框架名和一组用于描述框架各方面具体属性的槽(Slot)组成。每个槽设有一个槽名,它的值描述框架所表示的事物的各组成部分的属性。在较复杂的框架中,槽下面还可进一步区分为多个侧面(Facet),每个侧面又有一个或多个侧面值,每个侧面值又可以是一个值或是一个概念的陈述。

框架的结构可以抽象地表示如下:

```
<框架名>
<槽名1> <侧面值11><值111>.....
 <侧面值12><值121>.....
 :
 :
 <侧面值1m><值1m1>.....
 :
 :
<槽名1> <侧面值 n1><值 n11>.....
 <侧面值 n2><值 n21>.....
 :
 :
 <侧面值 nm><值 nm1>.....
```

以下所示是一个描述椅子概念的简单框架的例子。该框架含有四个槽:物体的范畴(建立物体间的属性继承关系)、椅子腿的数目、靠背式样和槽值。

CHAIR Frame

```
Specialization-of: FURNITURE
Number-of-Legs: an integer (DEFAULT 4)
Style-of-Back: Straight, cushioned
Number-of-Arms: 0, 1 or 2
```

(a)

JOHN'S - CHAIR Frame

```
Specialization-of: CHAIR
Number-of-Legs: 4
Style-of-Back: Cushioned
Number-of-Arms: 0
```

(b)

下面的示例是关于饭店的一个框架结构。

Generic RESTAURANT Frame

Specialization-of: Business—Establishment

Types:

range: ( Cafeteria, Seat-Yourself, Wait-To-Be-Seated)

default: Wait-To-Be-Seated

if-needed: IF plastic-orange-counter THEN Cafeteria

IF wait-for-Waitress-sign OR reservations-made

THEN wait-To-Be-Seated

OTHERWISE Seat-Yourself

Location:

range: an ADDRESS

if-needed: (Look at the MENU)

Name:

if-needed: (Look at the MENU)

Food-Style:

Range: (Burgers, Chinese, American, Seafood, French)

default: American

if-added: (Update Alternatives of Restaurant)

Times-of-Operation:

Range: a Time-of-Day

default: open evenings except Mondays

Payment-Form:

Range: (Cash, Credit-Card, Check)

Alternatives:

Range: all Restaurants with same Food-Style

if-needed: (Find all Restaurants with the same Food-Style)

由上述两个例子可见，在框架知识表示中，除了表示框架各种属性的槽或侧面，还经常使用默认(DEFAULT)值侧面和附加过程(IF-NEEDED)侧面。

一个槽的默认侧面为槽的属性值提供了一个隐含值。如椅子的腿通常为4个，如果一个实际问题的上下文没有提供相反的证据，则认为隐含值是正确的。

一个槽的附加过程侧面，包含一个附加过程，在上下文和默认侧面都没有给出需要的属性值时，附加过程给出槽值的计算过程或填槽时要做的动作，通常对应于一组子程序。在关于饭店的框架中，其 Types, Location, Name, Food-Style 和 Alternatives 各槽都有附加过程侧面。正是这种附加过程，把过程性知识有机地结合到框架的表示之中。

综上所述，在框架系统中，每个侧面有四种填写方式：(1) 靠已知的情况或物体属性提供；(2) 通过默认隐含；(3) 通过调用框架的继承关系实现属性值继承；(4) 对附加过程侧面通过执行附加过程实现。

在框架系统的框架之间，除有继承关系外，还可能具有嵌套关系。如程序中的 Location 槽，是关于地址的描述，这样的“地址”概念本身有可能是另一个框架结构，从而形成了框架的嵌套。

## 二、框架系统的推理

对一个给定的问题,框架推理主要完成两种推理活动:一是匹配,即根据已知事实寻找合适的候选框架;二是填槽,即填写候选框架中的未知槽值,从而寻找出未被给出或尚未发现的事实。

### (一) 匹配

当利用由框架所构成的知识库进行推理,形成概念和作出决策时,其过程往往是根据已知的信息,通过和知识库中预先存储的框架进行匹配,即逐槽比较,从中找出一个或几个与该信息所提供的情况最适合的候选框架。然后再对所有候选框架进行评估,以决定最合适的选择框架。这些评估准则通常很简单,如以某个或某些重要属性是否存在,某属性值是否属于允许的误差范围等为条件来判定匹配是否成立。较复杂的评估准则可以是一组产生式规则或过程,用来推导匹配是否成功。在实际构造框架系统时,可以根据特定应用领域的要求来定义合适的判定原则。

如果当前候选框架匹配失败,这就需要选择其他的候选框架。从失败的候选框架中有可能得到一些下一个应选框架的线索,这种线索使得控制转换到另一个更有可能的候选框架中,从而不必放弃以前的全部工作而一切从头开始。为此,有以下几种方法可以尝试。

1. 找出当前候选框架中已经匹配成功的框架片段,把这个框架片段同其他在同一层次上的可能候选框架进行片段匹配。如果匹配成功,则当前候选框架中的许多属性值可以填入新的候选框架。

2. 在框架中建立另一个专门的槽,这类槽中存放一些本框架匹配不成功时应转向哪个方向进行试探的建议,这些建议能使系统的控制转移到另外的框架上去。例如,在图5-19(a)中,可以建立一个 MOVE 槽,存入“如果没有靠背并且太宽,则建议用 BENCH(长凳)框架;如果太高并且没有靠背,则建议用 STOOL(凳子)框架”。

3. 沿着框架系统排列的层次向上回溯。如从狗框架→哺乳动物框架→动物框架,直到找到一个足够通用,且不与已知信息矛盾的框架。

### (二) 填槽

推理过程中填槽的方式有四种:查询、默认、继承和附加过程计算。其中查询方式是指使用系统先前推理中得出的,仍得留在当前数据库中的中间结果或者由系统之外的用户输入到当前数据库的数据。默认和继承方式是相对简单的填槽方式,因为它们不需要系统做过多的推理,这种特性是框架表示有效性的一个重要方面。它使得框架推理可以使用根据以往经验得到的属性值,而无须重新计算。附加过程计算的推理方式使得框架系统的问题求解通过特定领域的知识而增强了求解效率。

## 三、框架表示的特点与不足

框架是一种经过组织的结构化知识表示方法。每个框架形成一个独立的知识单元,其上的操作相对独立,从而使框架表示有较好的模块性,便于扩充。框架表示对知识的描述模拟了人

脑对事物的多方面、多层次的存储结构，直观自然，易于理解、且充分反映事物间内在的联系。框架表示中的附加过程侧面使框架不但能描述静态知识，而且还能反映过程性知识，而且把两者有机地融合在一起，形成一个整体系统。

框架表示的不足在于框架结构本身还没有形成完整的理论体系，框架、槽和侧面等各知识表示单元缺乏清晰的语义，其表达知识的能力尚待增强，支持其应用的工具尚待开发。此外，在多重继承时有可能产生多义性，如何解决继承过程中概念属性的歧义，目前尚没有统一的方法。

因而，框架系统适合于表示典型的概念、事件和行为。在一些大型的系统中，框架表示的使用总是与其他模式（如产生式系统）有机地结合在一起。

### 第三节 面向对象的表示方法

把面向对象(OBJECT)的方法和技术用于知识表示和处理具有极好的发展前景。面向对象方法的本质是强调从客观世界中固有的事物出发来构造系统；强调系统中的对象及对象之间的关系能够如实地反映客观世界中的固有事物及其关系。用面向对象方法构造的知识系统能够比较自然地反映人们思考问题的方式，人在认知和分析问题时，总是把问题分解为一些对象及对象之间的组合和联系。事实上，任何系统都可以视作为达到某种目的而相互作用的一组对象组成。

#### 一、关于对象的定义

一个对象的形式定义可以用以下四元组表示：

对象 $::= <ID, DS, MS, MI>$

其中 ID 是对象标识符，DS 是对象的数据结构，MS 是对象的方法集合。

对象的标识符 ID(Identifier)又称对象名，用以标识一个特定的对象，就像人有人名，学校有校名，军队有番号，任一特定事物都有特定标记一样。

对象的数据结构 DS(DataStructure)描述了对象当前的内部状态或所具有的静态属性，常用一组 $<\text{属性名 属性值}>$ 表示。例如，一支军队当前所在的地理位置、各级首长的情况、当前的兵力配备和武器装备状况等都可以作为该支军队(对象)的内部状态。

对象的方法集合 MS(Method Set)用以说明对象所具有的内部处理方法或对受理的消息的操作过程，它反映了对象自身的智能行为。例如，军人以服从命令为天职，对于上级的一个命令或决定如何具体贯彻执行，需要制定有关措施；一个军事指挥员，根据上级意图、我情、敌情和天时地利，对作战方案应作出合理的决策。在面向对象的系统中，这些措施和决策都是通过方法描述的。又如，在化学中，碳在一定条件下遇到氧会变成一氧化碳或二氧化碳；在物理学中，U<sup>235</sup>在一定条件下会发生裂变，释放大量能量；在算术中，一个自然数可以与另一个自然数进行加、减、乘、除等运算。这里的化学反应过程、物理反应过程和算术运算法则分别属碳、U<sup>235</sup>和自然数等三类对象所具有的方法。

对象的消息接口 MI(Message Interface)是对象接收外部信息和驱动有关内部方法的唯一对外接口。这里的外部信息称为消息。例如，一支军队接收的消息可能有上级的命令、友邻部队的请求、敌军的撤退或进攻信息，以及其他各种军事情报等。发送消息的对象称为发送者，接收

消息的对象称为接收者。消息接口以消息模式集的形式给出,每一消息模式有一消息名,通常还包含必要的参数表。当接收者从它的消息接口受理发送者的某一消息时,首先要判断该消息属哪一消息模式,找出与之匹配的内部方法,然后,执行与该消息相联的方法,进行相应的消息处理或回答某些信息。

在面向对象的系统中,问题求解或程序的执行是依靠对象间传递消息完成的。最初的消息通常来自用户的输入,某一对象在处理相应的消息时,如果需要,又可以通过传递消息去请求其他对象完成某些处理工作或回答某些信息,其他对象在执行所要求的处理时同样可以通过传递消息与别的对象联系,如此下去,直至得到问题的解。

## 二、消息、接口和方法

通常而言,消息(Message)是指在通信双方之间传送的任何书面、口头或代码的内容。在面向对象的方法中,对对象实施操作的唯一途径就是向对象发送消息,各对象间的联系只有通过消息发送和接收来进行。这里的“消息”具体化为对对象执行某一处理或回答某一请求的信息。

向对象发送消息就是要求该对象根据消息使用其相应的处理能力。某一对象在执行相应的处理时,如果需要的话,该对象可以通过发送消息请求其他对象完成某些处理工作或回答信息,其他对象在执行某些请求时,同样可以通过消息发送与别的对象联系。发送者发送消息,它仅仅告诉消息接受者做什么,至于如何做,则完全由接受者决定。消息发送者只知道消息接受者具有某种功能,而不知道它是如何实现的。消息接受者独立地决定采用什么方式完成所需处理,对接受到的消息进行分析并作出相应的反应,或改变自身的内部状态或向发送者回答消息。向对象发送消息实质上是一个间接的过程调用,即驱动对象中同消息选择库所指明的操作相对立的过程。在大多数情况下,这类调用会引起该对象向其他对象发送新的消息,这种消息发送一直延续到某个基本操作为止,终止消息发送链。这种消息发送过程如图5-19所示。

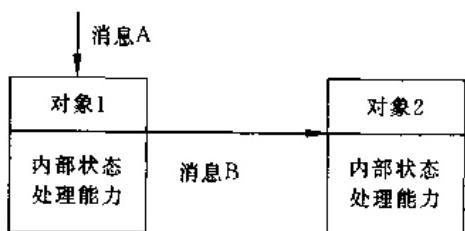


图5-19 对象间的消息传递

同一消息可以送往不同的对象,不同的对象对于相同形式的消息,可以有不同的解释,不同的反应;一个对象可以接受不同形式、不同内容的多个消息。把对象可响应的消息集称为该对象同系统其他部分的消息接口,它是对象间进行相互作用的唯一接口。每个消息所对应的处理过程是由对象内部定义的一种方法(method)实现的。通常情况下,对象的消息接口应尽可能地通用,以便对象能够尽可能多地响应消息。

## 三、类(class)

在面向对象的程序设计中,那些具有相同结构和处理能力的对象用类来描述。一个类实质上定义了一种对象类型,它描述了属于该对象类型的所有对象的性质。如果一个对象具有某个类所描述的特性,则该对象就是这个类的异构实例(instance)。例如,如果把“整数”这个概念比作类的话,那么“4”便可看作该类的一个实例。

在类的描述中,通常要包括类名(CLASS NAME),类变量(CLASS VAR),实例变量(IN-  
• 88 •

STANCE VAR)及一组称为方法的操作程序。类变量指出这类对象，其值为属于该类的所有对象所共享，其相对应的实例变量则用来说明这类对象各自的内部特性。通常同一类对象其实例变量的值不一定相同，只有附加于对象本身的方法才能访问这些变量。方法部分则是由一组操作程序组成，形成对象的处理能力。例如，可以用一个类 POINT 描述平面上所有的点：

类名：POINT

类变量：无

实例变量：X；0(点的横坐标，初值为0)

Y；0(点的纵坐标，初值为0)

COLOR；RED(点的颜色；初值为红色)

操作程序：

| 消息选择符        | 对应的过程名即定义          | 过程所需变量  | 注释     |
|--------------|--------------------|---------|--------|
| MOVE-X       | MOVE-POINT-X       | 点的新的横坐标 | 水平移动点  |
| MOVE-Y       | MOVE-POINT-Y       | 点的新的纵坐标 | 垂直移动点  |
| CHANGE-COLOR | CHANGE-POINT-COLOR | 点的新颜色   | 改变点的颜色 |
|              |                    | :       |        |

例如，平面上位于(3,5)处一个红色的点，可用对象 POINT1 描述于下：

POINT1

内部状态 点的 X 坐标：3  
点的 Y 坐标：5  
点的颜色：红色

能受理的消息(处理能力) 处理能力的具体表现

MOVE-X 与消息 MOVE-X 相对应过程的定义  
MOVE-Y 与消息 MOVE-Y 相对应过程的定义  
CHANGE-COLOR 与消息 CHANGE-COLOR 相对应过程的定义

上例给出的对象 POINT1 则可看作为类 POINT 的一个实例。类的描述中变量部分对应于对象的内部状态，操作程序部分对应于对象的处理能力。

对象是在系统运行的过程中由其所属的类动态生成的。在一些系统中，生成新的对象是通过向该对象所属的类发送一个特殊的消息完成的。一个类可以生成多种不同的对象，这些对象具有相同的结构和处理能力，但内部状态可以各不相同。对象一经建立便具有类的描述中的所有特性，人们可以通过消息接口向对象发送消息进行相应的操作。例如，对象 POINT1 现位于(0,0)点，若要将它移到(20,0)处，则可通过函数 SEND 完成：

(SEND POINT1 "MOVE-X" 20)

其中参数 POINT1 为消息接收者，MOVE-X 为消息选择符，20 则为相应过程的实参。对象 POINT1 接到消息后，即驱动与该消息选择符相对应的过程 MOVE-POINT-X，并以 20 作为该过程的实参，完成对 POINT1 的移动。操作完成后，POINT1 的内部过程发生了变化，其 X 的值

变成了20。

在面向对象的程序设计中,类与类之间形成一种层次结构,图5-20即为这种结构的描述。

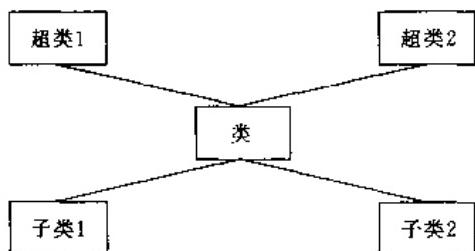


图5-20 类的层次结构

一个类可以有超类(Super-Class),也可以有子类(Sub-Class)。比较抽象、通用的类靠近这种层次结构的顶部,而较具体、专用的类则靠近其底部。这与人类习惯的从宏观到微观刻画事物的方法是一致的。类的层次结构的一个重要特性是继承,一个类可以直接继承其超类中所描述的全部特性,这种继承还具有传递性。因此,一个类的对象除了具有该类所描述的特性外,还具有层次结构中位于其上层所有父辈超类所描述的全部特性。

## 四、封装与继承

封装与继承是面向对象方法的两个重要特征。

### (一) 封装

简单而言,所谓封装(Encapsulation)是每个对象内部的数据及对数据的操作不允许其他对象直接引用和修改。对象内部状态和方法对外界都是隐蔽的。于是,每个对象就像集成电路和芯片一样被封装在一个明确的范围内,对外接口是它的消息模式集,受“黑盒”保护的内部实现由它的状态和操作细节组成。

一个复杂对象常由若干相对简单的对象组成。简单对象所提供的某些消息有时可能仅供复杂对象内部使用,复杂对象的这种不向外界公开的消息称为复杂对象的私有消息。在日常生活中,有许多关于私有消息的例子。例如,一个人或一个集团有多种多样的能力,虽然许多能力是可以公开告知外界的,但也有些纯属内部或不愿对外公开和不提供对外服务的能力。这些不对外公开的能力所对应的消息就是私有消息。相对地,对象向外界公开提供的消息称为该对象的公有消息。

封装使对象的设计者与对象的使用者分开,使用者不用知道对象行为的实现细节而只需通过该对象的消息接口便可访问该对象。明确地把该对象的外部定义和对象的内部实现分开是面向对象系统的一个特色,封装性本身就是模块性,模块的定义和实现分开,使面向对象的知识系统便于维护和修改。

### (二) 继承

继承是一种表示类之间的相似性的机制,在定义一个先前已经定义的类相似的类时能简化类的定义工作。已存在的类可称为新建立的类的超类;新建立的类则可称为已存在的类的子类。

继承机制可有层次继承与复合继承两种。在层次继承中,一个类最多只能有一个超类,子类中的对象最多只能直接继承其超类中所描述的特性,但可以有多个属于同一个超类。在子类描述中可以对超类中已有的描述进行修改,例如为某个变量赋予新的初值,对某个消息选择符定义新的过程,或在子类中增加新的描述(如定义新变量和操作程序等)。例如前面已定义了一

个类 POINT 描述平面上的点,为了描述平面上的直线,可以利用继承性定义类 POINT 的一个子类 LINE。由于描述一条直线只须指定直线的起点、方向及长度,因此,在类 LINE 中只须定义变量 DIR 和 LEN 来描述直线的方向和长度,以及一组相应操作程序,而直线的起点这个变量则可从前面定义过的类 POINT 中继承。同时类 LINE 还可以继承类 POINT 中所描述的对点的操作程序,如 MOVE-X,MOVE-Y 等,以下便是类 LINE 的描述。

类名: LINE

类变量: 无

实例变量: DIR; 0(直线和水平轴的夹角,初值为0)

LEN; 0(直线的长度,初值为0)

超类: POINT

操作程序:

| 消息选择符      | 相应的过程名及定义       | 所需变量  | 注释      |
|------------|-----------------|-------|---------|
| CHANGE-DIR | CHANGE-LINE-DIR | 直线的角度 | 改变直线的方向 |
| :          |                 |       |         |
| CHANGE-LEN | CHANGE-LINE-LEN | 直线的长度 | 改变直线的长度 |
| :          |                 |       |         |

这里,“超类:POINT”指明了类 LINE 的超类是类 POINT,这样属于类 LINE 的对象除了具有上述类 LINE 中描述的特性外,还继承了类 POINT 中所描述的全部特性。因此在类 LINE 的描述中,无须再定义如直线的起点以及相应的操作等,减少了冗余信息。

再来看一个日用品的例子,它的层次结构如图5-21所示。

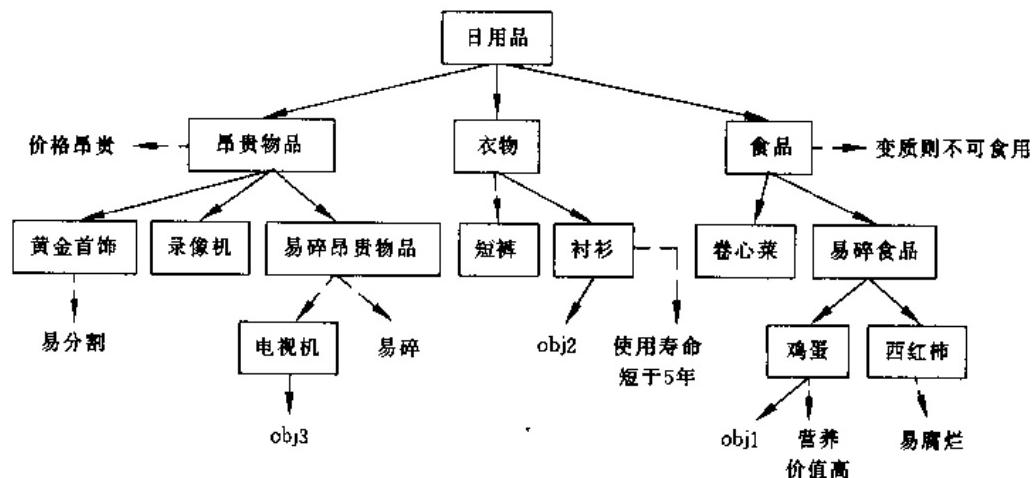


图 5-21 一个日用品的层次结构图

图中“<类1>→<类2>”表示<类2>是<类1>的一个子类,“<类> <特性>”表示<类>具有<特性>。这里,食品、衣物及昂贵物品被组织为日用品的子类,而这些类又有自己的子类,如衬衫、短裤是衣物的子类,而易碎食品、卷心菜则是食品的子类等。最下边的一行为具体的对象:obj1,obj2,obj3,它们分别表示具体的一只鸡蛋、一件衬衫和一台电视机。由于在这种层次结构中,较通用的类靠近层次结构的顶部,而较专用的类则靠近层次结构的底部,类的继承按照自底向上的原则,即较低层次类的对象可继承较高层次类中所描述的特性。例如对象 obj1 可以从鸡蛋、易碎食品、食品及日用品这四个类中继承它们的全部特性:营养价值较高、易

碎、如变质则不可食用等。

除了这种层次继承外,有些支持面向对象的程序设计系统还提供复合继承机制。同层次继承的不同之处是:一个类可以有一个以上的超类。这样就可以把若干个类的描述结合到一起,因此,可更大程度地提高信息的共享。如上例中,有两个类的对象具有“易碎”这个特性,如易碎食品、易碎昂贵物品等,虽然有些冗余,如果能利用复合继承机制,使一个类可以有一个以上的超类,便能够更好地实现信息共享。我们只需建立一个新类:易碎日用品,专门用于描述那些易碎的日用品。这样,鸡蛋、电视机这两个类便可直接从易碎日用品这个类中继承“易碎”这个特性,而无须建立易碎食品及易碎昂贵物品这两个类了。利用复合继承机制,前面的日用品框架可由图5-22表示。

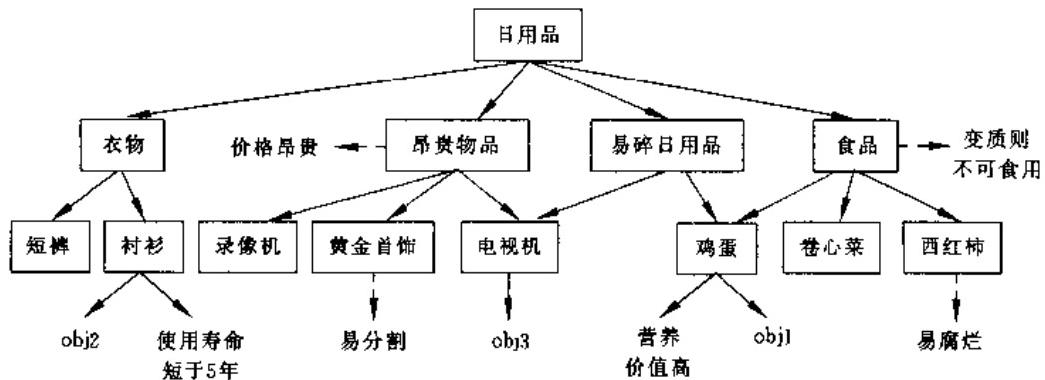


图5-22 改进后的日用品层次结构图

## 五、面向对象知识表示与语义网络、框架系统的比较

语义网络、框架系统和面向对象知识表示都是结构化的知识表示方法,面向对象知识表示是其中最结构化的方法。

语义网络的主要优点是灵活性,网络中的结点和有向弧可以按规定不加限制地定义。这种灵活性在面向对象的表示中不仅仍然存在,而且对象和对象之间的关系还可以动态地建立。其中,语义网络中的结点对应于面向对象中的对象,语义网络中由有向弧定义的语义联系对应于面向对象结构中的消息传播,而语义网络中的实例联系、泛化联系对应于面向对象方法中对象、子类与类之间的继承关系。事实上,面向对象的结构可以看成一种动态的语义网络。

语义网络的主要缺点是难于进行系统的开发和维护。随着对象(结点)的增加,语义网络的管理变得十分复杂,很难给出某个对象和属性值的修改对整个系统的影响。面向对象方法的封装性恰恰有力地克服了语义网络的这一弱点。

框架结构与面向对象的结构很类似,知识都可以使用类的概念按一定的层次结构来组织。然而,框架知识表示的模块性不能清楚地定义。削弱框架系统结构化的两大因素是:框架之间的关系既可能是具有继承关系的子类连接和成员连接,也可能是反映全体和部分关系的组成连接,而且不是唯一的;规则可以通过一个框架连接到另一个框架。在面向对象的知识表示中,两个类之间的连接关系只有子类连接,而规则用作一个类的内部方法,不可能出现跨越两个对象的规则,类的唯一对外接口是消息模式,类之外的代码唯有通过传递有关消息才能与该类的方法打交道。因此,面向对象的表示特别适合于大型知识系统的开发和维护。

综上所述,面向对象程序设计方法,以信息隐蔽和抽象数据类型概念为基础,既提供了从一般到特殊的演绎手段(如继承等),又提供了从特殊到一般的归纳形式(如类等),已成为主宰90年代软件(特别是基于知识的人工智能软件)的开发方法。

目前,面向对象技术不仅在软件设计和知识表示中得到了广泛的应用,而且已拓广到数据库管理和多媒体系统等新的应用领域。与此同时,出现了一个比“对象”更富动态性、更具有人工智能含义的概念,即智能体(agent)。智能体是一种可以包含主观信念和承诺,且可以主动响应外界事件的对象。可以认为,面向智能体技术是面向对象技术在人工智能与专家系统应用领域中的发展。

## 习 题 五

1. 语义网络、框架系统和面向对象的知识表示的要点是什么?它们有何联系和区别?
2. 用语义网络表示下列知识:
  - (1) 我是一个人。
  - (2) 我拥有我的计算机。
  - (3) 我的计算机的拥有者是我。
  - (4) 我的计算机是 PC/P I 400。
  - (5) PC/P I 400是微机。
  - (6) 微机是计算机。
  - (7) PC/P I 400包括硬盘、显示器、微处理器、内存。
  - (8) 硬盘、显示器、微处理器、内存是 PC/P I 400的组成部分。
3. 用语义网络分别表示下列命题:
  - (1) 如果车库起火,那么用二氧化碳或沙来扑灭。
  - (2) 所有的学生都用计算机算题。
4. 说明对象、实例变量和类之间的区别。
5. 用框架系统来描述旅馆房间的概念,包括房间结构和基本设施两方面。房间结构包括墙壁、门的数量、地面的材料和颜色;基本设施包括椅子、电话、床有关内容,对于床还需特别说明关于垫子的有关内容。

## 第六章 不确定知识表示及推理

在日常生活和科学的研究中，人们一直在追求用某一确定的数学模型或 Contor 集合理论来解决问题或描述现象。然而，真实世界是不断变化的、不完全的、不精确的，充满着矛盾的、复杂相关的信息世界。这就是说，用确定的概念描述事物往往是有局限性的。正如著名的逻辑学家 Russell 所说的“所有的传统逻辑都习惯地假设所使用的符号是精确的，所以它就不能适用于我们这个人间的世界，而只能适应于一个理想中的天堂……，逻辑研究比别的任何研究都使我们更接近上帝。”

在现实世界中，包含有大量的柔性信息，表征出模糊性、复杂性和不精确性，因而不精确推理、非单调推理和模糊推理就变得十分重要了。

在这一章，将讨论几种基本的不确定知识表示及推理的方法。

### 第一节 不确定推理概述

当代冯·诺依曼计算机具有强大的计算能力，在数值计算、数据处理、文档处理、工业控制乃至知识工程中的逻辑推理方面发挥出来的本领已远远超过人类。不过，计算机在这些应用领域中所解决的问题都是“良性设定问题”，即求解问题的前提条件明确，求解的算法存在，并且可以用某种程序设计语言进行明确地描述。然而，在真实的世界中，存在有大量求解算法难以描述的非良性设定问题需要解决。一般说来，问题的不确定性来自知识的客观现实和对知识的主观认识水平。现实世界中，几乎没有事情是完全确定的，处理不确定性问题的目的是希望得到对某一命题的尽可能精确的判断。对于不确定性推理来说，不确定性的描述和不确定性的传播是两个主要问题。

#### 一、不确定性问题的代数模型

一个问题的代数模型是由论域、运算和公理三部分所组成的。所以，我们讨论各种不确定性推理方法时应符合某种代数结构。比如，在某种意义上构成半群，即在不确定性值域上，不确定性度量的合成运算具有封闭性并满足结合律。按照这种方法建立起来的不确定性问题模型需要涉及下面的三个问题。

##### (一) 不确定性知识的表示

不确定性知识的表示主要解决用什么方法来描述知识的不确定性问题。常用的方法有数值法和非数值法。数值法以概率方法、确定因子法、D-S 证据理论和可能性理论为代表；非数值法则以批注理论和非单调逻辑为代表。数值法表示便于计算、比较，非数值法表示便于定性分析，两种方法的结合是描述不确定性知识的好办法。

## (二) 不确定性知识的推理

不确定性知识的推理是指知识不确定性的传播和更新,即新的不确定性知识的获取过程。这个过程是在“公理”(比如领域专家给出的规则强度和用户给出的原始证据的不确定性度)的基础上,定义一组函数,计算出“定理”(非原始数据的命题)的不确定性度量。也就是说,根据原始证据的不确定性和知识的不确定性,求出结论的不确定性。

一个不确定性知识推理包括如下的算法:

算法1: 根据规则前提  $E$  的不确定性  $C(E)$  和规则强度  $f(H, E)$ , 求出假设  $H$  的不确定性  $C(H)$ , 即定义函数  $g_1$ , 使得

$$C(H) = g_1[C(E), f(H, E)]$$

算法2: 根据分别由独立的证据  $E_1$  和  $E_2$  所求得的假设  $H$  的不确定性  $C_1(H)$  和  $C_2(H)$ , 求出证据  $E_1$  和  $E_2$  的组合所导致的假设  $H$  的不确定性  $C(H)$ , 即定义函数  $g_2$ , 使得

$$C(H) = g_2[C_1(H), C_2(H)]$$

算法3: 根据两个证据  $E_1$  和  $E_2$  的不确定性  $C(E_1)$  和  $C(E_2)$ , 求出证据  $E_1$  和  $E_2$  的合取的不确定性, 即定义函数  $g_3$ , 使得

$$C(E_1 \wedge E_2) = g_3[C(E_1), C(E_2)]$$

算法4: 根据两个证据  $E_1$  和  $E_2$  的不确定性  $C(E_1)$  和  $C(E_2)$ , 求出证据  $E_1$  和  $E_2$  的析取的不确定性, 即定义函数  $g_4$ , 使得

$$C(E_1 \vee E_2) = g_4[C(E_1), C(E_2)]$$

观察图6-1所示的推理网络。设  $A_1, A_2, A_3$  和  $A_4$  为原始证据, 即已知证据  $A_1, A_2, A_3$  和  $A_4$  的不确定性分别为  $C(A_1), C(A_2), C(A_3)$  和  $C(A_4)$ 。求  $A_5, A_6$  和  $A_7$  的不确定性。

问题的求解过程为:

1. 由证据  $A_1$  和  $A_2$  的不确定性  $C(A_1)$  和  $C(A_2)$ , 根据算法4求出  $A_5$  和  $A_2$  析取的不确定性  $C(A_1 \vee A_2)$ 。

2. 由  $A_1$  和  $A_2$  析取的不确定性  $C(A_1 \vee A_2)$  和规则  $R_1$  的规则强度  $f_1$ , 根据算法1求出  $A_5$  的不确定性  $C(A_5)$ 。

3. 由证据  $A_3$  和  $A_4$  的不确定性  $C(A_3)$  和  $C(A_4)$ , 根据算法3求出  $A_3$  和  $A_4$  合取的不确定性  $C(A_3 \wedge A_4)$ 。

4. 由  $A_3$  和  $A_4$  合取的不确定性  $C(A_3 \wedge A_4)$  和规则  $R_2$  的规则强度  $f_2$ , 根据算法1求出  $A_6$  的不确定性  $C(A_6)$ 。

5. 由  $A_5$  的不确定性  $C(A_5)$  和规则  $R_3$  的规则强度  $f_3$ , 根据算法1求出  $A_7$  的其中的一个不确定性  $C'(A_7)$ 。

6. 由  $A_6$  的不确定性  $C(A_6)$  和规则  $R_4$  的规则强度  $f_4$ , 根据算法1求出  $A_7$  的另外一个不确定性  $C''(A_7)$ 。

7. 由  $A_7$  的两个根据独立证据分别求出的不确定性  $C'(A_7)$  和  $C''(A_7)$ , 根据算法2求出  $A_7$  最后的不确定性  $C(A_7)$ 。

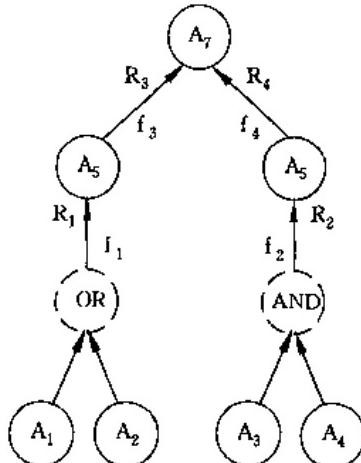


图6-1 不确定推理网络示意图

## (三) 不确定推理的语义

综上所述,对于一个不确定推理问题应指出不确定性表示和推理的含义。基于概率论的方法较好地解决这个问题。

如规则强度  $f(B, A)$  可理解为当证据  $A$  为真时, 对假设  $B$  为真的一种影响程度; 而  $C(A)$  可理解为  $A$  为真的程度, 即对于  $f(B, A)$  和  $C(A)$ , 应给出:

对于  $f(B, A)$  而言:

- (1)  $A$  为真则  $B$  为真, 这时  $f(B, A) = ?$
- (2)  $A$  为真则  $B$  为假, 这时  $f(B, A) = ?$
- (3)  $A$  对  $B$  没有影响时, 这时  $f(B, A) = ?$

对于  $C(A)$  而言:

- (1)  $A$  为真时,  $C(A) = ?$
- (2)  $A$  为假时,  $C(A) = ?$
- (3) 对  $A$  一无所知时,  $C(A) = ?$

## 二、几种主要的不确定性推理方法

不确定性推理方法是 70 年代提出并开展研究的。比如 Shortliffe 等人 1975 年在 MYCIN 专家系统开发中, 提出了确定因子法; Duda 等人 1976 年在 PROSPECTOR 专家系统开发中给出了主观贝叶斯法; Dempster 和 Shafer 1976 年提出了证据理论; Zadeh 在 1978 年提出了可能性理论, 1983 年又提出了模糊理论; Cohen 在 1985 年也论述了一种称为批注理论的非数值方法等。下面简要地介绍这几种方法。

### 1. 确定因子法

确定因子法是 MYCIN 专家系统中使用的不确定性推理方法。该方法以确定性理论为基础, 采用可信度来刻画不确定性。其优点是简单、实用, 在许多专家系统中得到了应用, 取得了较好的效果。

### 2. 主观贝叶斯方法

主观贝叶斯方法是 PROSPECTOR 专家系统中使用的不确定性推理方法。它是基于贝叶斯 (Bayes) 公式修正后而形成的一种不确定性推理方法。该方法的优点是具有较强的数学基础, 计算工作量也较为适中。

### 3. D-S 证据理论

D-S 证据理论是由 Dempster 提出, 由他的学生 Shafer 发展起来的。该理论引进了信任函数, 这些函数可以满足比概率函数的公理还要弱的公理, 因而可以用来处理由“不知道”所引起的不确定性。

### 4. 可能性理论

可能性理论的基础是 Zadeh 本人的模糊集合理论。正如概率论处理的是由随机性引起的不确定性一样, 可能性理论处理的是由模糊性引起的不确定性。

### 5. 批注理论

批注理论 (Endorsement) 是一种非数值方法。它将系统所使用的推理规则和议程中的任务都加以批注。规则的批注提出前提条件与规则结论的关系, 任务的批注指出该任务的结论与议程中另一任务的结论之间的协同、冲突、潜在冲突及冗余情况。这些批注与数据源、数据类型和数据的精度有关。该理论的优点是可以表示出用数值难以表达的较复杂的关系。缺点是系统每步规则推理都要将前提的批注转移到结论中, 从而使得结论中的批注迅速增长, 对于结论的选择变得困难。

## 第二节 不确定推理方法

下面详细讨论几种主要的不确定推理方法。

### 一、确定性因子法

#### (一) 知识的不确定性

在 MYCIN 中,知识是由规则表示的。它的一般形式为:

IF  $E_1$  AND  $E_2$  AND …… AND  $E_n$   
THEN  $H(x)$

其中  $E_i (i=1, 2, \dots, n)$  是证据,  $H$  可以是一个或多个结论。具有此规则形式的解释为当证据  $E_1, E_2, \dots, E_n$  都存在时, 结论  $H$  具有  $x$  确定性因子 CF(Certainty Factor)。 $x$  的具体值由领域专家主观地给出,  $x$  的取值范围为  $[-1, 1]$  内。 $x > 0$  表示证据存在, 增加结论为真的确定性程度,  $x$  越大结论越真,  $x = 1$  表示证据存在结论为真。相反,  $x < 0$  表示证据存在, 增加结论为假的确定性程度,  $x$  越小结论越假,  $x = -1$  表示证据存在结论为假。 $x = 0$  时, 则表示证据与结论无关。

在 MYCIN 中, 系统引入了信任增长度 MB(Measure Belief) 和不信任增长度 MD(Measure Disbelief) 两个概念, 用来描述知识的不确定性。

$MB(H, E) > 0$  表示因证据  $E$  的出现而增加对假设  $H$  为真的信任增长度, 即  $P(H|E) > P(H)$ ; 而  $MD(H, E) > 0$  表示因证据  $E$  的出现而增加对假设  $H$  为真的不信任增长度, 即  $P(H|E) < P(H)$ 。为此, MB 和 MD 的定义为

$$MB(H, E) = \begin{cases} 1 &; \text{当 } P(H) = 1 \text{ 时} \\ \frac{\max\{P(H|E), P(H)\} - P(H)}{1 - P(H)} &; \text{其他} \\ 0 &; \text{当 } P(H) = 0 \text{ 时} \end{cases}$$
$$MD(H, E) = \begin{cases} 1 &; \text{其他} \\ \frac{\min\{P(H|E), P(H)\} - P(H)}{-P(H)} &; \text{当 } P(H) = 0 \text{ 时} \end{cases}$$

在 MYCIN 中, 还使用了另一个基本概念——确定性因子 CF, 它的作用是把 MB 和 MD 组合在一起, CF 的定义为:

$$CF(H, E) = MB(H, E) - MD(H, E)$$

根据 MB、MD 和 CF 的定义, 我们可以导出如下的性质:

#### 1. MB 与 MD 的互斥性

当  $MB(H, E) > 0$  时,  $MD(H, E) = 0$

当  $MD(H, E) > 0$  时,  $MB(H, E) = 0$

MB 与 MD 的互斥性表明对于同一个证据  $E$ , 不可能同时既增加对  $H$  的信任增长度, 又增加对  $H$  的不信任增长度, 即不可能因为证据  $E$  的出现, 既使  $P(H|E) > P(H)$ , 又使  $P(H|E) < P(H)$ 。

#### 2. 假设的互斥性

若对于同一证据有  $n$  个互不相容的假设  $H_i (i=1, 2, \dots, n)$ , 则

$$\sum_{i=1}^n \text{CF}(H_i | E) \leq 1$$

仅当证据  $E$  在逻辑上蕴含某个假设  $H_i$  时, 上式的等式才成立。

根据性质(1)可以直接用概率值来表示:

$$\text{CF}(H, E) = \begin{cases} \frac{P(H|E) - P(H)}{1 - P(H)} & ; \text{ 若 } P(H|E) > P(H) \\ 0 & ; \text{ 若 } P(H|E) = P(H) \\ \frac{P(H|E) - P(H)}{P(H)} & ; \text{ 若 } P(H|E) < P(H) \end{cases}$$

### 3. 值域

MB、MD 和 CF 的值域为:

$$0 \leq \text{MB}(H, E) \leq 1$$

$$0 \leq \text{MD}(H, E) \leq 1$$

$$-1 \leq \text{CF}(H, E) \leq 1$$

### 4. 证实的情况

$$\text{CF}(H|E) = \begin{cases} -1 & ; \text{ 若 } P(H|E) = 0 \\ 0 & ; \text{ 若 } P(H|E) = P(H) \\ 1 & ; \text{ 若 } P(H|E) = 1 \end{cases}$$

### 5. 不证实的情况

若  $\text{MB}(H, E) = 0$ , 说明  $E$  的存在证实不了  $H$ , 或者是  $E$  与  $H$  独立, 或者是  $E$  否认  $H$ 。

若  $\text{MD}(H, E) = 0$ , 说明  $E$  的存在不否认  $H$ , 或者是  $H$  与  $E$  独立, 或者是  $E$  证实  $H$ 。

若  $\text{CF}(H|E) = 0$ , 表示  $E$  与  $H$  独立。

### 6. CF 不同于概率 $P$

从上面的定义可以看出, 确定性因子法中的 CF 与概率  $P$  有一定的对应关系, 但又有所区别, 因为对于概率恒有:

$$P(H|E) + P(\sim H|E) = 1$$

而

$$\text{CF}(H|E) + \text{CF}(\sim H|E) = 0$$

由此可见, 确定性因子法的 CF 的概念与概率  $P$  的概念是不同的。

## (二) 证据的不确定性

在 MYCIN 系统中, 证据的不确定性是用证据的确定性因子  $\text{CF}(E)$  表示的。原始证据的确定性因子由用户主观地给出, 非原始证据的确定性因子由不确定性推理获得。

证据的不确定性有下面的几个性质:

### 1. 值域

当证据  $E$  以某种程度为真时, 有  $0 \leq \text{CF}(E) \leq 1$ 。

当证据  $E$  以某种程度为假时, 有  $-1 \leq \text{CF}(E) < 0$ 。

### 2. 典型值

当证据  $E$  肯定为真时, 有  $\text{CF}(E) = 1$ 。

当证据  $E$  肯定为假时, 有  $\text{CF}(E) = -1$ 。

当证据  $E$  一无所知时,有  $CF(E)=0$ 。

### (三) 不确定性推理算法

#### 1. $E$ 肯定存在的情况

在证据  $E$  肯定存在时有  $CF(E)=1$ ,那么结论  $H$  的确定性因子为真,即有

$$CF(E)=CF(H,E)$$

#### 2. $E$ 不是肯定存在的情况

在客观的现实世界中,对证据的观察往往也是不确定的。除此之外,证据  $E$  可能还是另一条规则的结论,这时也常常是不确定的。在这种情况下,结论  $H$  的确定性因子  $CF(H)$  不仅取决于规则的确定性因子  $CF(H,E)$ ,而且还取决于证据  $E$  的确定性因子  $CF(E)$ 。计算公式为

$$CF(H)=CF(H,E)\cdot\max\{0,CF(E)\}$$

#### 3. 证据是多个条件的逻辑组合的情况

我们分三种情况进行讨论。

##### (1) 证据是合取连接

若系统有规则形如

$$\text{IF } E_1 \text{ AND } E_2 \text{ AND } \dots \text{ AND } E_n \text{ THEN } H(x),$$

那么,有  $CF(E)=CF(E_1 \text{ AND } E_2 \text{ AND } \dots \text{ AND } E_n)$

$$=\min\{CF(E_1), CF(E_2), \dots, CF(E_n)\}$$

##### (2) 证据是析取连接

这时, $E=E_1 \text{ OR } E_2 \text{ OR } \dots \text{ OR } E_n$ ,有

$$CF(E)=CF(E_1 \text{ OR } E_2 \text{ OR } \dots \text{ OR } E_n)$$

$$=\max\{CF(E_1), CF(E_2), \dots, CF(E_n)\}$$

##### (3) 两条规则具有相同结论的情况

若有两条规则分别是

$$\text{IF } E_1 \text{ THEN } H(CF(H,E_1))$$

$$\text{IF } E_2 \text{ THEN } H(CF(H,E_2))$$

那末首先分别计算出  $CF_1(H)$  和  $CF_2(H)$ :

$$CF_1(H)=CF(H,E_1)\cdot\max\{0,CF(E_1)\}$$

$$CF_2(H)=CF(H,E_2)\cdot\max\{0,CF(E_2)\}$$

然后用公式

$$CF_{12}(H)=\begin{cases} CF_1(H)+CF_2(H)-CF_1(H)\cdot CF_2(H); & \text{若 } CF_1(H)\geq 0 \text{ 且 } CF_2(H)\geq 0 \\ CF_1(H)+CF_2(H)+CF_1(H)\cdot CF_2(H); & \text{若 } CF_1(H)<0 \text{ 且 } CF_2(H)<0 \\ CF_1(H)+CF_2(H); & \text{其他} \end{cases}$$

计算出由  $E_1$  和  $E_2$  组合而导出的确定性因子  $CF_{12}(H)$ 。

例 6-1 下面通过一个例子说明该方法的推理过程。

例如有如下的推理规则:

Rule 1: IF  $E_1$  THEN  $H(0.9)$

Rule 2: IF  $E_2$  THEN  $H(0.7)$

Rule 3: IF  $E_3$  THEN  $H(-0.8)$

Rule 4: IF  $E_4 \text{ AND } E_5$  THEN  $E_1(0.7)$

Rule 5: IF  $E_6$  AND( $E_7$  OR  $E_8$ ) THEN  $E_2(1.0)$

规则形成的推理网络如图6-2所示。

在图6-2中,  $E_3, E_4, E_5, E_6, E_7$  和  $E_8$  为原始证据, 其确定性因子由用户给出, 假定它们的值为:

$$CF(E_3) = 0.3, \quad CF(E_4) = 0.9, \quad CF(E_5) = 0.6,$$

$$CF(E_6) = 0.7, \quad CF(E_7) = -0.3, \quad CF(E_8) = 0.8$$

系统的求值顺序是先求出  $CF(E_1)$ 、 $CF(E_2)$  和  $CF(E_3)$ , 然后再求  $CF(H)$ 。

$$\begin{aligned} CF(E_1) &= 0.7 \times \max\{0, CF(E_4 \text{ AND } E_5)\} \\ &= 0.7 \times \max\{0, \min\{CF(E_4), CF(E_5)\}\} \\ &= 0.7 \times \max\{0, \min\{0.9, 0.6\}\} \\ &= 0.7 \times \max\{0, 0.6\} \\ &= 0.7 \times 0.6 \\ &= 0.42 \end{aligned}$$

$$\begin{aligned} CF(E_2) &= 1 \times \max\{0, CF(E_6 \text{ AND } (E_7 \text{ OR } E_8))\} \\ &= 1 \times \max\{0, \min\{CF(E_6), \max\{CF(E_7), CF(E_8)\}\}\} \\ &= 1 \times \max\{0, \min\{CF(E_6), \max\{-0.3, 0.8\}\}\} \\ &= 1 \times \max\{0, \min\{0.7, 0.8\}\} \\ &= 1 \times \max\{0, 0.7\} \\ &= 1 \times 0.7 \\ &= 0.7 \end{aligned}$$

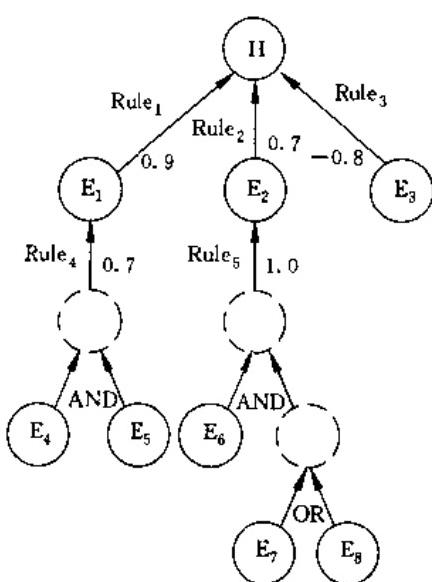


图6-2 规则形成的推理网络

$$\begin{aligned} CF(E_3) &= 0.3 \\ CF_1(H) &= 0.9 \times \max\{0, CF(E_1)\} \\ &= 0.9 \times \max\{0, 0.42\} \\ &= 0.9 \times 0.42 \\ &= 0.38 \end{aligned}$$

$$\begin{aligned} CF_2(H) &= 0.7 \times \max\{0, CF(E_2)\} \\ &= 0.7 \times \max\{0, 0.7\} \\ &= 0.7 \times 0.7 \\ &= 0.49 \end{aligned}$$

$$\begin{aligned} CF_3(H) &= -0.8 \times CF(E_3) \\ &= -0.8 \times 0.3 \\ &= -0.24 \\ \because CF_1(H) > 0 \text{ 且 } CF_2(H) > 0 \\ \therefore CF_{12}(H) &= CF_1(H) + CF_2(H) \\ &\quad - CF_1(H) \times CF_2(H) \\ &= 0.38 + 0.49 - 0.38 \times 0.49 = 0.6838 \end{aligned}$$

又  $\because CF_{12}(H) > 0, CF_3(H) < 0$

$$\begin{aligned} \therefore CF_{123}(H) &= CF(H) = CF_{12}(H) + CF_3(H) \\ &= 0.6838 - 0.24 = 0.4438 \end{aligned}$$

所以由此推出结论  $H$  的确定性因子  $CF(H) = 0.4438$ 。

确定性因子法的优点是：

1. 简单、直观。主要表现在组合假设和证据的不确定性的计算十分简单，而且不需要把信任和不信任的判断知识表示成概率的形式。我们知道，把某个值指派给一个假设的确定性因子要比把一个概率直接指派给一个假设要容易得多。

2. 计算仅有线性的信息和时间的复杂度，而且推理的近似效果也比较理想。

确定性因子法的缺点是：

1. 缺乏较强的数学基础作为自己的理论支撑。

2. 在使用该方法的过程中，会发现一旦一个不支持假设的证据影响可以导致抑制多个支持该假设的证据的影响，反之亦然。

3. 如果多个规则与一个规则在逻辑上是等价的，比如规则  $A \rightarrow B, C \rightarrow B, D \rightarrow B$  与规则  $(A \wedge (C \vee D)) \rightarrow B$  在逻辑上是等价的。此时，采用一个规则和采用多个规则计算  $B$  的 CF 值就不相同，因为后者可能导致计算的累计误差。

4. 组合规则的使用顺序不同，可能得到完全不同的结果。

5. 当证据不存在时，证据对假设的影响就无法确定，即该方法无法将不知道和不确定区分开来。

## 二、主观 Bayes 方法

主观 Bayes 方法是最早用于处理不确定性推理的方法之一。在专家系统的不确定性推理中，知识工程师使用 Bayes 定理的形式来描述不确定性知识。Bayes 定理可叙述为

若有诸事件  $A_1, A_2, \dots, A_n$ ，两两互斥，事件  $B$  为事件  $A_1 + A_2 + \dots + A_n$  的子事件，且  $P = (A_i) > 0 (i=1, 2, \dots, n), P(B) > 0$ ，那么按乘法定理有

$$P(B)P(A_i|B) = P(A_i)P(B|A_i)$$

所以

$$P(A_i|B) = \frac{P(A_i)P(B|A_i)}{P(B)}$$

又由全概率公式，可得到

$$P(B) = P(A_1)P(B|A_1) + P(A_2)P(B|A_2) + \dots + P(A_n)P(B|A_n)$$

因此有

$$P(A_i|B) = \frac{P(A_i)P(B|A_i)}{P(A_1)P(B|A_1) + P(A_2)P(B|A_2) + \dots + P(A_n)P(B|A_n)} \\ (i=1, 2, \dots, n)$$

我们称这个公式为 Bayes 公式，同时称  $P(A_1), P(A_2), \dots, P(A_n)$  的值为先验概率； $P(A_1|B), P(A_2|B), \dots, P(A_n|B)$  的值为后验概率。Bayes 公式就是从先验概率推导出后验概率的公式。

### (一) 规则不确定性的描述

在基于规则的专家系统中，Bayes 公式可以写成为

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

在直接使用 Bayes 公式描述不确定性时，计算后验概率  $P(H|E)$  需要知道先验概率  $P(E|$

$H$ ),而因为先验概率又是难以给出的量,为了克服这一困难,所以就提出了主观 Bayes 方法。下面我们介绍主观 Bayes 方法对规则不确定性的描述。

先给出几个定义:

### 1. 几率函数

几率函数定义为  $O(x) = \frac{P(x)}{1-P(x)}$ 。它表示证据  $x$  的出现概率与不出现概率之比,从定义可知,随着  $P(x)$  的增大, $O(x)$  也增大,而且

当  $P(x)=0$  时,有  $O(x)=0$

和 当  $P(x)=1$  时,有  $O(x)=\infty$

这样,取值为  $[0,1]$  的  $P(x)$  被放大为取值为  $[0,\infty]$  的  $O(x)$ 。

### 2. 充分性度量

充分性度量定义为

$$LS = \frac{P(E|H)}{P(E|\sim H)}$$

### 3. 必要性度量

必要性度量定义为

$$LN = \frac{P(\sim E|H)}{P(\sim E|\sim H)}$$

对于证据确定的情况,此时  $P(E)=1$  或  $P(E)=0$ ,现在假定有规则:

IF  $E$  THEN  $H$

根据 Bayes 公式有

$$\begin{aligned} P(H|E) &= P(E|H)P(H)/P(E) \\ P(\sim H|E) &= P(E|\sim H)P(\sim H)/P(E) \end{aligned}$$

将上述两式相除,得

$$\frac{P(H|E)}{P(\sim H|E)} = \frac{P(E|H)}{P(E|\sim H)} \cdot \frac{P(H)}{P(\sim H)}$$

再利用几率函数和 LS,上式可表示为

$$O(H|E) = LS \cdot O(H)$$

从这个式子可以看出,LS 越大, $O(H|E)$  就越大,而  $P(H|E)$  越大,说明  $E$  对  $H$  的支持越强。当  $LS \rightarrow \infty$  时, $O(H|E) \rightarrow \infty$ ,从而有  $P(H|E) \rightarrow 1$ ,说明  $E$  的存在导致  $H$  为真。因此,我们说  $E$  对  $H$  是充分的,且称 LS 为充分性度量。

同理,可以得到 LN 反映了  $\sim H$  的出现对  $H$  的支持程度。当  $LN=0$  时,将导致  $O(H|\sim E)=0$ ,说明  $E$  不存在导致  $H$  为假。因此,我们说  $E$  对  $H$  是必要的,且称 LN 为必要性度量。

在主观 Bayes 方法中,一条规则变成了如下的形式:

IF  $E$  THEN(LS,LN)  $H$

其中参数 LS,LN 和先验几率  $O(H)$  要由领域专家主观给出,先计算出后验几率,再计算出后验概率。

## (二) 证据不确定性的描述

在主观 Bayes 方法中,证据的不确定性是采用概率  $P$  的等价形式——几率函数  $O(x)$  来描述的。下面就证据  $E$  确定和不确定时的情况分别进行讨论。

1. 当证据  $E$  确定必出现时, 可直接使用公式

$$O(H|E) = LS \cdot O(H)$$

和

$$O(H|\sim E) = LN \cdot O(H)$$

以求得使用规则  $E \rightarrow H$  后,  $O(H)$  的更新值  $O(H|E), O(H|\sim E)$ 。若需要以概率的形式表示, 再由公式

$$P(E) = \frac{O(E)}{1+O(E)}$$

计算出  $P(H|E)$  和  $P(H|\sim E)$ 。

2. 当证据  $E$  不确定时, 即  $P(E) \neq 1$ , 这时需要作如下的工作。

设  $E'$  是与  $E$  有关的所有观察, 对规则  $E \rightarrow H$  来说有公式(1976年由 Duda 给出)

$$P(H|E') = P(H|E) \cdot P(E|E') + P(H|\sim E) \cdot P(\sim E|E')$$

当  $P(E|E')=1$  时, 证据  $E$  必然出现, 有

$$P(H|E') = P(H|E) = \frac{LS \cdot P(H)}{(LS-1) \cdot P(H) + 1}$$

我们不难验证此公式的正确性。

证明:

$$\begin{aligned} \frac{LS \cdot P(H)}{(LS-1) \cdot P(H) + 1} &= \frac{\frac{P(E|H)}{P(E|\sim H)} \cdot P(H)}{\left[ \frac{P(E|H)}{P(E|\sim H)} - 1 \right] \cdot P(H) + 1} \\ &= \frac{P(E|H) \cdot P(H)}{P(E|\sim H) - P(H) \cdot P(E|\sim H) + P(H) \cdot P(E|H)} \\ &= \frac{\frac{P(E) \cdot P(H|E)}{P(H)}}{\frac{P(E) \cdot P(\sim H|E)}{P(\sim H)} [1 - P(H)] + \frac{P(E) \cdot P(H|E)}{P(H)} \cdot P(H)} \\ &= \frac{P(E) \cdot P(H|E)}{P(E) \cdot P(\sim H|E) + P(E) \cdot P(H|E)} \\ &= \frac{P(H|E)}{P(H|E) + P(\sim H|E)} \\ &= P(H|E) \end{aligned}$$

证毕。

当  $P(E|E')=0$  时, 证据  $E$  必然不出现, 同理

有

$$P(H|E') = P(H|\sim E) = \frac{LN \cdot P(H)}{(LN-1) \cdot P(H) + 1}$$

当  $P(E|E')=P(E)$  时, 即观察  $E'$  对  $E$  无影响,

有

$$\begin{aligned} P(H|E') &= P(H|E) \cdot P(E) + P(H|\sim E) \cdot P(\sim E) \\ &= P(H) \end{aligned}$$

这样, 我们可以从  $P(E|E')$  分别为 0,  $P(E)$  和 1 确定与其相应的  $P(H|E')$  值。主观 Bayes 方法采用分段线性插值的方法, 由  $P(E|E')$  的任意取值, 可得到相应的  $P(H|E')$  值, 如图 6-3 所示。

### 3. 证据合取情况

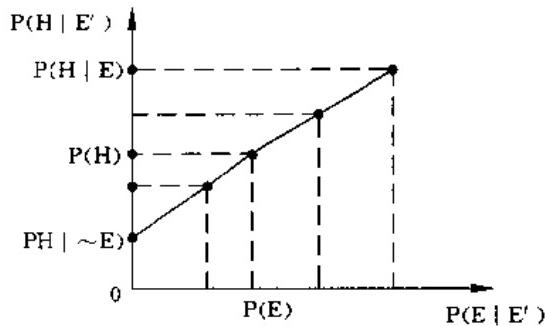


图 6-3 分段线性插值图

设在观察  $E'$  之下, 证据  $E_1, E_2, \dots, E_n$  的概率为  $P(E_1 | E'), P(E_2 | E'), \dots, P(E_n | E')$ , 那么有

$$P(E_1 \text{ AND } E_2 \text{ AND } \dots \text{ AND } E_n | E') = \min\{P(E_1 | E'), P(E_2 | E'), \dots, P(E_n | E')\}$$

#### 4. 证据析取情况

设在观察  $E'$  之下, 证据  $E_1, E_2, \dots, E_n$  的概率为  $P(E_1 | E'), P(E_2 | E'), \dots, P(E_n | E')$ , 那么有

$$P(E_1 \text{ OR } E_2 \text{ OR } \dots \text{ OR } E_n | E') = \max\{P(E_1 | E'), P(E_2 | E'), \dots, P(E_n | E')\}$$

#### 5. 相互独立的证据导出同一假设情况

例 6-2 设一组相互独立的证据  $E_1, E_2, \dots, E_n$  的观察分别为  $E'_1, E'_2, \dots, E'_n$ , 并且有规则  $E_1 \rightarrow H, E_2 \rightarrow H, \dots, E_n \rightarrow H$ 。假定由这些规则得到的假设  $H$  的后验几率分别是  $O(H | E'_1), O(H | E'_2), \dots, O(H | E'_n)$ , 那么由这些独立证据的组合相应得到的假设  $H$  的后验几率为

$$O(H | E'_1 \& E'_2 \& \dots \& E'_n) = \frac{O(H | E'_1)}{O(H)} \cdot \frac{O(H | E'_2)}{O(H)} \cdots \frac{O(H | E'_n)}{O(H)} \cdot O(H)$$

### (三) 举例

例 6-3 PROSPECTOR 专家系统中的部分推理网络如图 6-4 所示。图中各结点的先验概率标在结点的右上方, 规则的 LS 和 LN 值标在该规则连线的一侧。用户给出的各原始证据在各自的观察之下概率为:

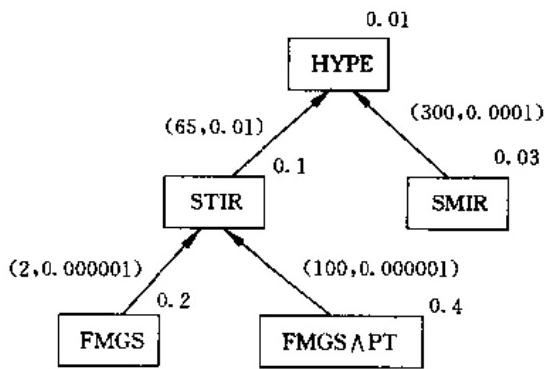


图 6-4 PROSPECTOR 中部分推理网络

$P(FMGS | S_1) = 0.7, P(FMGS \wedge PT | S_2) = 0.6, P(STIR | S_3) = 0.02$ 。现要求计算假设 HYPE 的后验概率  $P(HYPE | S_1 \wedge S_2 \wedge S_3)$ 。

解 1. 根据  $P(\text{FMGS}|S_1)$  计算  $P(\text{STIR}|S_1)$

由于  $P(\text{FMGS}|S_1) = 0.7 > 0.2$ , 所以

$$P(\text{STIR}|\text{FMGS}) = \frac{\text{LS} \cdot P(\text{STIR})}{(\text{LS}-1) \cdot P(\text{STIR}) + 1} = \frac{2 \times 0.1}{(2-1) \times 0.1 + 1} \\ = \frac{0.2}{1.1} = 0.1818$$

$$P(\text{STIR}|S_1) = P(\text{STIR}) + \frac{P(\text{STIR}|\text{FMGS}) - P(\text{STIR})}{1 - P(\text{FMGS})} \\ \times [P(\text{FMGS}|S_1) - P(\text{FMGS})] \\ = 0.1 + \frac{0.1818 - 0.1}{1 - 0.2} (0.7 - 0.2) \\ = 0.151125$$

2. 根据  $P(\text{FMGS} \wedge \text{PT}|S_2)$  计算  $P(\text{STIR}|S_2)$ 。

由于  $P(\text{FMGS} \wedge \text{PT}|S_2) = 0.6 > 0.4$ , 所以

$$P(\text{STIR}|\text{FMGS} \wedge \text{PT}) = \frac{\text{LS} \cdot P(\text{STIR})}{(\text{LS}-1)P(\text{STIR}) + 1} = \frac{100 \times 0.1}{99 \times 0.1 + 1} \\ = \frac{10}{10.9} = 0.9174311$$

$$P(\text{STIR}|S_2) = P(\text{STIR}) + \frac{P(\text{STIR}|\text{FMGS} \wedge \text{PT}) - P(\text{STIR})}{1 - P(\text{FMGS} \wedge \text{PT})} \\ \times [P(\text{FMGS} \wedge \text{PT}|S_2) - P(\text{FMGS} \wedge \text{PT})] \\ = 0.1 + \frac{0.9174311 - 0.1}{1 - 0.4} \cdot (0.6 - 0.4) \\ = 0.372477$$

3. 根据独立证据 FMGS 和  $\text{FMGS} \wedge \text{PT}$  计算  $P(\text{STIR}|S_1 \wedge S_2)$ 。

先计算后验几率  $O(\text{STIR}|S_1 \wedge S_2)$ , 由于

$$O(\text{STIR}) = \frac{P(\text{STIR})}{1 - P(\text{STIR})} = \frac{0.1}{1 - 0.1} = 0.111111$$

$$O(\text{STIR}|S_1) = \frac{P(\text{STIR}|S_1)}{1 - P(\text{STIR}|S_1)} = \frac{0.151125}{1 - 0.151125} = 0.1780297$$

$$O(\text{STIR}|S_2) = \frac{P(\text{STIR}|S_2)}{1 - P(\text{STIR}|S_2)} = \frac{0.372477}{1 - 0.372477} = 0.593567$$

由此得

$$O(\text{STIR}|S_1 \wedge S_2) = \frac{O(\text{STIR}|S_1)}{O(\text{STIR})} \cdot \frac{O(\text{STIR}|S_2)}{O(\text{STIR})} \cdot O(\text{STIR}) \\ = \frac{0.1780297}{0.111111} \cdot \frac{0.593567}{0.111111} \cdot 0.111111 \\ = 0.9510532$$

然后计算后验概率  $P(\text{STIR}|S_1 \wedge S_2)$ , 得

$$P(\text{STIR}|S_1 \wedge S_2) = \frac{O(\text{STIR}|S_1 \wedge S_2)}{1 + O(\text{STIR}|S_1 \wedge S_2)} = \frac{0.9510532}{1.9510532} \\ = 0.4874563$$

4. 根据  $P(\text{STIR}|S_1 \wedge S_2)$  计算  $P(\text{HYPE}|S_1 \wedge S_2)$ 。

由于  $P(\text{STIR}|S_1 \wedge S_2) = 0.4874563 > 0.1$ , 所以

$$P(\text{HYPE}|\text{STIR}) = \frac{LS \cdot P(\text{HYPE})}{(LS-1)P(\text{HYPE})+1} = \frac{65 \times 0.01}{64 \times 0.01 + 1} = 0.3963414$$

$$P(\text{HYPE}|S_1 \wedge S_2) = P(\text{HYPE}) \frac{P(\text{HYPE}|\text{STIR}) - P(\text{HYPE})}{1 - P(\text{STIR})} \times [P(\text{STIR}|S_1 \wedge S_2) - P(\text{STIR})]$$

$$= 0.01 + \frac{0.3963414 - 0.01}{1 - 0.1} \cdot (0.4874563 - 0.1)$$

$$= 0.1763226$$

5. 根据  $P(\text{SMIR}|S_3)$  计算  $P(\text{HYPE}|S_3)$ 。

由于  $P(\text{SMIR}|S_3) = 0.02 < 0.03$ , 所以

$$P(\text{HYPE}|\sim\text{SMIR}) = \frac{LN \cdot P(\text{HYPE})}{(LN-1) \cdot P(\text{HYPE}) + 1}$$

$$= \frac{0.0001 \times 0.01}{-0.9999 \times 0.1 + 1} = 0.000001$$

$$P(\text{HYPE}|S_3) = P(\text{HYPE}|\sim\text{SMIR}) + \frac{P(\text{HYPE}) - P(\text{HYPE}|\sim\text{SMIR})}{P(\text{SMIR})} \cdot P(\text{SMIR}|S_3)$$

$$= 0.000001 + \frac{0.01 - 0.000001}{0.03} \times 0.02$$

$$= 0.006667$$

6. 根据独立证据 STIR 和 SMIR 计算  $P(\text{HYPE}|S_1 \wedge S_2 \wedge S_3)$

先计算后验几率  $O(\text{HYPE}|S_1 \wedge S_2 \wedge S_3)$ , 由于

$$O(\text{HYPE}) = \frac{P(\text{HYPE})}{1 - P(\text{HYPE})} = \frac{0.01}{1 - 0.01} = 0.010101$$

$$O(\text{HYPE}|S_1 \wedge S_2) = \frac{P(\text{HYPE}|S_1 \wedge S_2)}{1 - P(\text{HYPE}|S_1 \wedge S_2)} = \frac{0.1763226}{0.8236774} = 0.2140675$$

$$O(\text{HYPE}|S_3) = \frac{P(\text{HYPE}|S_3)}{1 - P(\text{HYPE}|S_3)} = \frac{0.006667}{0.993333} = 0.00671174$$

因此得

$$O(\text{HYPE}|S_1 \wedge S_2 \wedge S_3) = \frac{O(\text{HYPE}|S_1 \wedge S_2) \cdot O(\text{HYPE}|S_3)}{O(\text{HYPE})} \cdot O(\text{HYPE})$$

$$= \frac{0.2140675 \cdot 0.00671174}{0.010101} \times 0.010101$$

$$= 0.142239$$

最后得到后验概率

$$P(\text{HYPE}|S_1 \wedge S_2 \wedge S_3) = \frac{O(\text{HYPE}|S_1 \wedge S_2 \wedge S_3)}{1 + O(\text{HYPE}|S_1 \wedge S_2 \wedge S_3)}$$

$$= \frac{0.142239}{1 + 0.142239} = 0.1245264$$

以上计算表明, 经推理后假设 HYPE 的概率已从先验概率 0.01 增大到后验概率 0.1245264。

主观 Bayes 方法的优点是:

1. 基于 Bayes 规则的计算方法具有公理基础和易于理解的数学性质, 它提供了两个规则强度, 恰当地处理了证据存在和不存在两种情况对假设的影响, 以及分段线性插值方法较好地处理了主观概率的数学不一致性。

2. 该方法的计算工作量适中。

主观 Bayes 方法的缺点是:

1. 要求具有指数级的先验概率, 可能导致组合爆炸。另外, 由于概率的分派具有主观性, 所

以在一个系统中很难保证由领域专家给出的概率具有前后的一致性。

2. 要求所有假设的概率都是独立的,这在一个大型的专家系统中,要求把解空间分解为相互排斥的子集可能是不实际的。

3. 在系统中增加或删除一个假设时,要对事件的概率进行修改。为了保证系统的相关性和一致性,还必须重新计算所有的概率。

4. 系统中的先验概率高度地依赖于上下文。例如,某种疾病的发生概率要依赖于地域位置和时间的变化,这给系统的处理带来困难。

### 三、D-S 证据理论

该理论提出的初期并没有引起人们的重视,直到80年代 Barnett(1981)、Friedman(1981)等人将这个方法应用于专家系统,才认识到它具有利用证据的积累可以缩小假设置信区间的重要优点。同时,D-S 证据理论能够处理由不知道而引起的不确定性。

#### (一) 证据的不确定性

设  $U$  表示所有可能假设的集合,且  $U$  中的各元素是相互排斥的。设  $U$  的元素个数为  $N$ ,则  $U$  的幂集合  $2^U$  中的元素个数为  $2^N$ ,每个幂集合的元素对应于一个关于假设取值的命题。同时,我们称  $U$  为辨别框(Frame of discernment)。

首先在  $U$  的幂集  $2^U$  上定义一个基本概率赋值函数(BPA, Basic Probability Assignment)  $m$ :

$$m: 2^U \rightarrow [0,1]$$

使  $m$  满足

$$m(\emptyset) = 0$$

$$\sum_{A \subseteq U} m(A) = 1$$

基本概率赋值函数  $m(A)$  表示证据对  $U$  的子集  $A$  成立的一种信任度量。 $m(A)$  的意义是:

(1) 若  $A \subset U$ , 且  $A \neq U$ , 则  $m(A)$  表示对  $A$  的确定信任程度。

(2) 若  $A = U$ , 则  $m(A)$  表示这个数不知如何分配。

(3) 若  $A \subseteq U$ , 且  $m(A) > 0$ , 则称  $A$  是  $m$  的一个焦元(Focal element)。

例如,设  $U = \{a, b, c\}$ , 在  $2^U$  上的基本概率赋值函数  $m$  为

$$m(\{\}, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}) = (0, 0.3, 0.1, 0, 0.2, 0.2, 0, 0.2)$$

其中  $m(\{a\}) = 0.3$  表示对命题  $\{a\}$  的确定信任程度为 0.3; 而  $m(\{a, b, c\}) = 0.2$  表示不知道 0.2 这个数如何分配。

值得注意的是  $m(\{a\}) + m(\{b\}) + m(\{c\}) = 0.3 + 0.1 + 0 = 0.4 \neq 1$ , 因此  $m(A)$  不是概率,因为概率函数  $P$  要满足  $P(a) + P(b) + P(c) = 1$ 。

在 D-S 证据理论中,信任函数 Bel 和似然函数 Pl 的概念起着重要作用。下面我们定义这两个函数。

信任函数 Bel 定义为

$$Bel: 2^U \rightarrow [0,1] \quad \text{且}$$

$$Bel(A) = \sum_{B \subseteq A} m(B)$$

即命题  $A$  的信任函数的值是  $A$  的所有子集的基本概率赋值函数  $m(B)$  ( $B \subseteq A$ ) 的数值和, 用信任函数  $\text{Bel}(A)$  表示对  $A$  的总信任程度。显而易见,  $\text{Bel}(\emptyset) = 0$ ,  $\text{Bel}(U) = 1$ 。在仅有单元素的集上,  $m(A)$  与  $\text{Bel}(A)$  是相等的。

似然函数  $\text{Pl}$  定义为

$$\text{Pl}: 2^U \rightarrow [0, 1] \quad \text{且}$$

$$\text{Pl}(A) = 1 - \text{Bel}(\sim A) = \sum_{B \cap A \neq \emptyset} m(B)$$

其中  $\sim A = U - A$ ,  $\text{Pl}(A)$  表示不否定  $A$  的信任程度, 它是所有与  $A$  相交子集的基本概率赋值函数  $m(B)$  ( $B \cap A \neq \emptyset$ ) 的数值和。

显然,  $\text{Bel}(A)$  和  $\text{Pl}(A)$  满足下面的关系:

$$\text{Pl}(A) \geq \text{Bel}(A)$$

$$\text{Pl}(\emptyset) = \text{Bel}(\emptyset) = 0$$

$$\text{Pl}(U) = \text{Bel}(U) = 1$$

$$\text{Pl}(A) = 1 - \text{Bel}(\sim A)$$

$$\text{Bel}(A) + \text{Bel}(\sim A) \leq 1$$

$$\text{Pl}(A) + \text{Pl}(\sim A) \geq 1$$

而  $\text{Pl}(A) - \text{Bel}(A)$  表示了既不信任  $A$  也不信任  $\sim A$  的一种度量, 即表示对  $A$  不知道的度量。

在 D-S 证据理论中, 由于缺少关于总概率的分配信息, 所以不能确切地知道概率是如何分配给每个元素  $x \in U$  的, 因而也就不可能计算与  $U$  的子集有关的概率  $P(A)$ 。这样, 我们就采取用区间  $(\text{Bel}(A), \text{Pl}(A))$  来描述  $A$  的不确定性。 $\text{Bel}(A)$  表示度量的下限,  $\text{Pl}(A)$  表示度量的上限, 即

$$\text{Bel}(A) \leq P(A) \leq \text{Pl}(A)$$

当  $(\text{Bel}(A), \text{Pl}(A)) = (1, 1)$  时, 因为此时  $\text{Bel}(A) = 1$ , 说明对  $A$  信任; 另一方面, 由于  $\text{Pl}(A) = 1$ , 即  $\text{Bel}(\sim A) = 1 - \text{Pl}(A) = 1 - 1 = 0$ , 说明对  $\sim A$  不信任。所以  $(\text{Bel}(A), \text{Pl}(A)) = (1, 1)$  表示  $A$  为真。

当  $(\text{Bel}(A), \text{Pl}(A)) = (0, 0)$  时, 因为  $\text{Bel}(A) = 0$ , 说明对  $A$  不信任; 另一方面, 由于  $\text{Bel}(\sim A) = 1 - \text{Pl}(A) = 1$ , 说明对  $\sim A$  信任。所以  $(\text{Bel}(A), \text{Pl}(A)) = (0, 0)$  表示  $A$  为假。

当  $(\text{Bel}(A), \text{Pl}(A)) = (0, 1)$  时, 因为  $\text{Bel}(A) = 0$ , 说明对  $A$  不信任; 另一方面, 由于  $\text{Bel}(\sim A) = 1 - \text{Pl}(A) = 0$ , 说明对  $\sim A$  也不信任。所以  $(\text{Bel}(A), \text{Pl}(A)) = (0, 1)$  表示对  $A$  一无所知。

当  $(\text{Bel}(A), \text{Pl}(A)) = (0.25, 1)$  时, 因为  $\text{Bel}(A) = 0.25$ , 而  $\text{Bel}(\sim A) = 1 - \text{Pl}(A) = 0$ , 所以表示对  $A$  为真有一定程度的信任, 0.25 表示对  $A$  为真的置信度。

当  $(\text{Bel}(A), \text{Pl}(A)) = (0, 0.75)$  时, 因为  $\text{Bel}(A) = 0$ , 而  $\text{Bel}(\sim A) = 1 - \text{Pl}(A) = 0.25$ , 所以表示对  $A$  为假有一定程度的信任。

当  $(\text{Bel}(A), \text{Pl}(A)) = (0.25, 0.65)$  时, 因为  $\text{Bel}(A) = 0.25$ , 说明对  $A$  为真有一定程度的信任; 另一方面,  $\text{Bel}(\sim A) = 1 - \text{Pl}(A) = 1 - 0.65 = 0.35$ , 表示对  $\sim A$  也有一定程度的信任。

由上述的讨论可知,  $\text{Bel}(A)$  表示对命题  $A$  为真的信任程度,  $\text{Bel}(\sim A)$  表示对  $\sim A$  的信任程度, 而  $\text{Pl}(A)$  表示对  $A$  为非假的信任程度, 而  $\text{Pl}(A) - \text{Bel}(A)$  则表示对  $A$  不知道的程度, 即既非对  $A$  信任又非不信任的那部分。

## (二) 证据的组合

对于相同的证据,由于来源不同,可能得不同的基本概率赋值函数,D-S 证据理论采用正交和来组合这些函数。

设  $m_1, m_2, \dots, m_n$  为  $2^U$  上的  $n$  个基本概率赋值函数,它们的正交和表示为  $m = m_1 \oplus m_2 \oplus \dots \oplus m_n$ ,且定义为

$$\begin{cases} m(\emptyset) = 0 \\ m(A) = K \sum_{\cap A_i = A} \prod_{1 \leq i \leq n} m_i(A_i), A \neq \emptyset \end{cases}$$

其中

$$K^{-1} = 1 - \sum_{\cap A_i = \emptyset} \prod_{1 \leq i \leq n} m_i(A_i) = \sum_{\cap A_i = \emptyset} \prod_{1 \leq i \leq n} m_i(A_i)$$

若  $K^{-1} = 0$ ,则说明  $m_i(A_i)$  之间是矛盾的。

例6-4 设  $U = \{a, b\}$ ,且从不同的证据源得知概率赋值函数分别为:

$$m_1(\emptyset, \{a\}, \{b\}, \{a, b\}) = (0, 0.4, 0.5, 0.1)$$

$$m_2(\emptyset, \{a\}, \{b\}, \{a, b\}) = (0, 0.6, 0.2, 0.2)$$

求正交和  $m = m_1 \oplus m_2$

解: 先求  $K^{-1}$

$$\begin{aligned} K^{-1} &= \sum_{x \cap y \neq \emptyset} m_1(x) \cdot m_2(y) \\ &= m_1(\{a\}) \cdot m_2(\{a\}) + m_1(\{a\}) \cdot m_2(\{a, b\}) + m_1(\{b\}) \cdot m_2(\{b\}) \\ &\quad + m_1(\{b\}) \cdot m_2(\{a, b\}) + m_1(\{a, b\}) \cdot m_2(\{b\}) + m_1(\{a, b\}) \cdot m_2(\{a, b\}) \\ &= 0.4 \times 0.6 + 0.4 \times 0.2 + 0.5 \times 0.2 + 0.5 \times 0.2 + 0.1 \times 0.6 \\ &\quad + 0.1 \times 0.2 + 0.1 \times 0.2 \\ &= 0.62 \end{aligned}$$

$$\text{再求 } m(A) = K \sum_{x \cap y = A} m_1(x) \cdot m_2(y)$$

$$\begin{aligned} m(\{a\}) &= \frac{1}{0.62} [m_1(\{a\}) \cdot m_2(\{a\}) + m_1(\{a, b\}) \cdot m_2(\{a\}) + m_1(\{a\}) \cdot m_2(\{a, b\})] \\ &= \frac{1}{0.62} [0.4 \times 0.6 + 0.1 \times 0.6 + 0.4 \times 0.2] \\ &= 0.61 \end{aligned}$$

$$\begin{aligned} m(\{b\}) &= \frac{1}{0.62} [m_1(\{b\}) \cdot m_2(\{b\}) + m_1(\{a, b\}) \cdot m_2(\{b\}) + m_1(\{b\}) \cdot m_2(\{a, b\})] \\ &= \frac{1}{0.62} [0.5 \times 0.2 + 0.1 \times 0.2 + 0.5 \times 0.2] \\ &= 0.36 \end{aligned}$$

$$\begin{aligned} m(\{a, b\}) &= \frac{1}{0.62} [m_1(\{a, b\}) \cdot m_2(\{a, b\})] \\ &= \frac{1}{0.62} [0.1 \times 0.2] \\ &= 0.03 \end{aligned}$$

所以有  $m(\emptyset, \{a\}, \{b\}, \{a, b\}) = (0, 0.61, 0.36, 0.03)$

## (三) D-S 证据理论的推理

### 1. 知识表示

设某领域的假设集合为  $U = \{s_1, s_2, \dots, s_n\}$ , 命题  $A, B, C \dots$  是  $U$  的子集, 系统的推理规则表示为

IF  $E$  THEN  $H, CF$

其中  $E$  为证据,  $H$  为假设, 它们是命题的逻辑组合,  $CF$  为置信度因子。

命题和置信度因子可表示为

$$A = \{a_1, a_2, \dots, a_k\}$$

$$CF = \{c_1, c_2, \dots, c_k\}$$

其中  $c_i$  用来描述  $a_i$  的置信度,  $i=1, 2, \dots, k$ 。

对任何命题  $A$ ,  $A$  的置信度  $CF$  应满足:

$$(1) c_i \geq 0, 1 \leq i \leq k$$

$$(2) \sum_{i=1}^k c_i \leq 1$$

## 2. 证据描述

设  $m$  为  $2^U$  上定义的基本概率赋值函数, 它应该满足下面的条件:

$$(1) m(\{s_i\}) \geq 0, \text{ 对于 } s_i \in U$$

$$(2) \sum_{i=1}^n m(\{s_i\}) \leq 1$$

$$(3) m(U) = 1 - \sum_{i=1}^n m(\{s_i\})$$

$$(4) m(A) = 0, \text{ 对于 } A \subseteq U, \text{ 且 } |A| > 1 \text{ 或 } |A| = 0 \text{ 其中 } |A| \text{ 表示命题 } A \text{ 中元素的个数。}$$

若  $m_1$  和  $m_2$  为  $2^U$  上的两个基本概率赋值函数, 则它们的正交和为

$$m(\{s_i\}) = \frac{1}{K} [m_1(\{s_i\}) \cdot m_2(\{s_i\}) + m_1(\{s_i\}) \cdot m_2(U) + m_1(U) \cdot m_2(\{s_i\})]$$

其中

$$K = m_1(U) \cdot m_2(U) + \sum_{i=1}^n [m_1(\{s_i\}) \cdot m_2(\{s_i\}) + m_1(\{s_i\}) \cdot m_2(U) + m_1(U) \cdot m_2(\{s_i\})]$$

如果  $K=0$ , 那么说明  $m_1$  和  $m_2$  矛盾。在  $K \neq 0$  的条件下, 定义信任函数  $Bel(A)$  为

对于任何命题  $A \subseteq U$ , 其信任函数为

$$Bel(A) = \sum_{B \subseteq A} m(B) = \sum_{a \in A} m(\{a\})$$

$$Bel(U) = \sum_{B \subseteq U} m(B) = \sum_{a \in U} m(\{a\}) + m(U) = 1$$

似然函数  $Pl(A)$  定义为

对于任何命题  $A \subseteq U$ , 其似然函数为

$$\begin{aligned} Pl(A) &= 1 - Bel(\sim A) = 1 - \sum_{a \in \sim A} m(\{a\}) \\ &= 1 - [\sum_{a \in U} m(\{a\}) - \sum_{a \in A} m(\{a\})] \\ &= 1 - [1 - m(U) - Bel(A)] \\ &= m(U) + Bel(A) \\ Pl(U) &= 1 - Bel(\sim U) \\ &= 1 - Bel(\emptyset) = 1 \end{aligned}$$

显然, 对于任何  $A \subseteq U$ , 都有

$$\text{Pl}(A) \geq \text{Bel}(A)$$

并且对于任何命题  $A \subseteq U, B \subseteq U$ , 都有

$$\text{Pl}(A) - \text{Bel}(A) = \text{Pl}(B) - \text{Bel}(B) = m(U)$$

在 D-S 证据理论中, 还利用  $\text{Pl}(A)$  和  $\text{Bel}(A)$  定义命题  $A$  的类概率函数  $f(A)$ , 作为命题确定性的度量。

设  $U$  为有限域, 对任何命题  $A \subseteq U$ , 命题  $A$  的类概率函数为

$$f(A) = \text{Bel}(A) + \frac{|A|}{|U|} [\text{Pl}(A) - \text{Bel}(A)]$$

其中  $|A|$  和  $|U|$  分别是  $A$  和  $U$  中元素的个数。易于证明, 类概率函数  $f(A)$  具有如下的性质:

$$(1) \sum_{a \in U} f(\{a\}) = 1$$

(2) 对于任何  $A \subseteq U$ , 有  $\text{Bel}(A) \leq f(A) \leq \text{Pl}(A)$

(3) 对任何  $A \subseteq U$ , 有  $f(\sim A) = 1 - f(A)$

由上述的性质, 还很容易得出下面的推论:

$$(1) f(\emptyset) = 0$$

$$(2) f(U) = 1$$

(3)  $0 \leq f(A) \leq 1$ , 对于任何  $A \subseteq U$ 。

例 6-5 设假设集合  $U = \{a, b, c\}$ , 基本概率赋值函数  $m$  为:

$$m(\emptyset, \{a\}, \{b\}, \{c\}, \{a, b, c\}) = (0, 0.5, 0.2, 0.2, 0.1)$$

若  $A = \{a, b\}$ , 试求  $\text{Bel}(A)$ ,  $\text{Pl}(A)$  和  $f(A)$ 。

解:

$$\begin{aligned} \text{Bel}(A) &= \text{Bel}(\{a, b\}) = m(\{a\}) + m(\{b\}) \\ &= 0.5 + 0.2 = 0.7 \end{aligned}$$

$$\begin{aligned} \text{Pl}(A) &= 1 - \text{Bel}(\sim \{a, b\}) = 1 - \text{Bel}(\{c\}) \\ &= 1 - m(\{c\}) \\ &= 1 - 0.2 = 0.8 \end{aligned}$$

$$\begin{aligned} f(A) &= \text{Bel}(A) + \frac{|A|}{|U|} [\text{Pl}(A) - \text{Bel}(A)] \\ &= 0.7 + \frac{2}{3} (0.8 - 0.7) = 0.77 \end{aligned}$$

### 3. 不确定推理

在下面的讨论中, 我们将所有输入的已知数据、条件部分和假设部分的命题都称为证据, 并且据此来分别讨论规则的条件部分和结论部分命题的确定性。

若  $A$  是规则条件部分的命题, 在证据  $E'$  的条件下, 命题  $A$  与证据  $E'$  的匹配程度定义为

$$\text{MD}(A, E') = \begin{cases} 1; & \text{如果 } A \text{ 的所有元素都出现在 } E' \text{ 中} \\ 0; & \text{其他} \end{cases}$$

那么, 规则的条件部分命题  $A$  的确定性为

$$\text{CER}(A) = \text{MD}(A, E') \cdot f(A).$$

由于  $f(A) \in [0, 1]$ , 所以有  $\text{CER}(A) \in [0, 1]$ 。

在规则的条件部分为若干个命题的逻辑组合的情况下, 整个条件部分的确定性是:

(1) 若  $A = A_1 \text{ AND } A_2 \text{ AND } \dots \text{ AND } A_n$ , 则

$$\begin{aligned} \text{CER}(A) &= \text{CER}(A_1 \text{ AND } A_2 \text{ AND } \dots \text{ AND } A_n) \\ &= \min\{\text{CER}(A_1), \text{CER}(A_2), \dots, \text{CER}(A_n)\} \end{aligned}$$

(2) 若  $A = A_1 \text{ OR } A_2 \text{ OR } \dots \text{ OR } A_n$ , 则

$$\begin{aligned} \text{CER}(A) &= \text{CER}(A_1 \text{ OR } A_2 \text{ OR } \dots \text{ OR } A_n) \\ &= \max\{\text{CER}(A_1), \text{CER}(A_2), \dots, \text{CER}(A_n)\} \end{aligned}$$

对于规则的假设部分的命题确定性,如果有规则 IF  $E$  THEN  $H = \{h_1, h_2, \dots, h_k\}$ , CF =  $\{c_1, c_2, \dots, c_k\}$ , 且  $U = \{h_1, h_2, \dots, h_k\}$ , 则  $U$  上的基本概率赋值函数为

$$m(\{h_1\}, \{h_2\}, \dots, \{h_k\}) = (\text{CER}(E) \cdot c_1, \text{CER}(E) \cdot c_2, \dots, \text{CER}(E) \cdot c_k)$$

$$m(U) = 1 - \sum_{i=1}^k [\text{CER}(E) \cdot c_i]$$

根据上述的基本概率赋值函数  $m$  就可以求出假设部分命题的信任函数、似然函数、类概率函数和确定性。

如果有几种规则支持同一命题时, 总的基本概率赋值函数  $m$  为各规则假设得到的基本概率赋值函数的正交和, 即

$$m = m_1 \oplus m_2 \oplus \dots \oplus m_n$$

#### 4. 实例

例6-6 在专家系统的推理过程中, 原始证据的确定性是由用户回答的。例如, 有如下的推理规则:

Rule 1 IF  $E_1 \text{ AND } E_2$  THEN  $A = \{a_1, a_2\}$ , CF = {0.3, 0.5}

Rule 2 IF  $E_3 \text{ AND } (E_4 \text{ OR } E_5)$  THEN  $N = \{n_1\}$ , CF = {0.7}

Rule 3 IF  $A$  THEN  $H = \{h_1, h_2, h_3\}$ , CF = {0.1, 0.5, 0.3}

Rule 4 IF  $N$  THEN  $H = \{h_1, h_2, h_3\}$ , CF = {0.4, 0.2, 0.1}

这些规则所形成的推理网络如图6-5所示。

设用户给出原始证据的确定性是:

$$\text{CER}(E_1) = 0.8, \text{CER}(E_2) = 0.6, \text{CER}(E_3) = 0.9, \text{CER}(E_4) = 0.5, \text{CER}(E_5) = 0.7$$

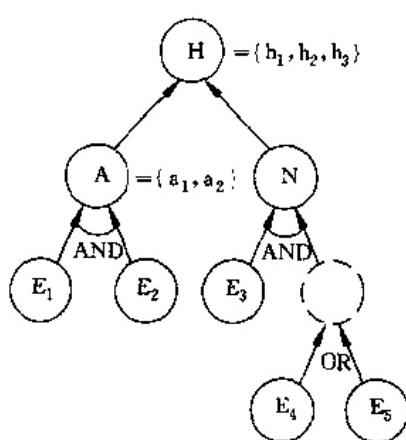


图6-5 例6-6的推理网络

并假定:  $|U| = 20$

解: 下面依次求出  $A$  的确定性  $\text{CER}(A)$ ,  $N$  的确定性  $\text{CER}(N)$  和  $H$  的确定性  $\text{CER}(H)$ 。

##### 1. 求 $\text{CER}(A)$

先求出 Rule 1 条件部分的确定性, 利用公式有

$$\begin{aligned} \text{CER}(E) &= \text{CER}(E_1 \text{ AND } E_2) \\ &= \min\{\text{CER}(E_1), \text{CER}(E_2)\} \\ &= \min\{0.8, 0.6\} = 0.6 \end{aligned}$$

再求假设  $A$  的基本概率赋值函数  $m$  为

$$\begin{aligned} m(\{a_1\}, \{a_2\}) &= (0.6 \times c_1, 0.6 \times c_2) \\ &= (0.6 \times 0.3, 0.6 \times 0.5) \\ &= (0.18, 0.3) \end{aligned}$$

于是有

$$\text{Bel}(A) = \sum_{a \in A} m(\{a\}) = 0.18 + 0.3 = 0.48$$

$$\text{Pl}(A) = 1 - \text{Bel}(\sim A) = 1 - 0 = 1$$

$$f(A) = \text{Bel}(A) + \frac{|A|}{|U|} [\text{Pl}(A) - \text{Bel}(A)]$$

$$=0.48 + \frac{2}{20}(1-0.48) = 0.53$$

$$\text{CER}(A) = \text{MD}(A, E') \times f(A) = 1 \times 0.53 = 0.53$$

其中  $\text{MD}(A, E') = 1$  是因为  $E'$  为已证实的命题(此例中为  $A$  的前提  $E_1$  和  $E_2$  均已证实)。

### 2. 求 $\text{CER}(N)$

利用公式,有

$$\begin{aligned}\text{CER}(E) &= \text{CER}(E_3 \text{ AND } (E_4 \text{ OR } E_5)) \\ &= \min\{\text{CER}(E_3), \max\{\text{CER}(E_4), \text{CER}(E_5)\}\} \\ &= \min\{0.9, \max\{0.5, 0.7\}\} \\ &= \min\{0.9, 0.7\} \\ &= 0.7\end{aligned}$$

$$m(\{n_1\}) = (0.7 \times c_1) = (0.7 \times 0.7) = 0.49$$

$$\text{Bel}(N) = \sum_{a \in A} m(\{a\}) = 0.49$$

$$\text{Pl}(N) = 1 - \text{Bel}(\sim N) = 1$$

$$\begin{aligned}f(N) &= \text{Bel}(N) + \frac{|N|}{|U|} [\text{Pl}(N) - \text{Bel}(N)] \\ &= 0.49 + \frac{1}{20}(1 - 0.49) = 0.52\end{aligned}$$

$$\text{CER}(N) = \text{MD}(N, E') \times f(N) = 0.52$$

### 3. 求 $\text{CER}(H)$

利用公式,有

$$\begin{aligned}m_1(\{h_1\}, \{h_2\}, \{h_3\}) &= (\text{CER}(A) \times c_1, \text{CER}(A) \times c_2, \text{CER}(A) \times c_3) \\ &= (0.53 \times 0.1, 0.53 \times 0.5, 0.53 \times 0.3) \\ &= (0.053, 0.265, 0.159)\end{aligned}$$

$$m_1(U) = 1 - (0.053 + 0.265 + 0.159) = 0.524$$

$$\begin{aligned}m_2(\{h_1\}, \{h_2\}, \{h_3\}) &= (\text{CER}(N) \times c_1, \text{CER}(N) \times c_2, \text{CER}(N) \times c_3) \\ &= (0.52 \times 0.4, 0.52 \times 0.2, 0.52 \times 0.1) \\ &= (0.208, 0.104, 0.052)\end{aligned}$$

$$m_2(U) = 1 - (0.208 + 0.104 + 0.052) = 0.636$$

然后求出  $m_1$  与  $m_2$  的正交和  $m_1 \oplus m_2$ 。为此,先求  $K^{-1}$

$$\begin{aligned}K^{-1} &= m_1(\{h_1\}) \cdot m_2(\{h_1\}) + m_1(\{h_1\}) \cdot m_2(\{U\}) + m_1(\{h_2\}) \cdot m_2(\{h_3\}) \\ &\quad + m_1(\{h_2\}) \cdot m_2(\{U\}) + m_1(\{h_3\}) \cdot m_2(\{h_3\}) + m_1(\{h_3\}) \cdot m_2(\{U\}) \\ &\quad + m_1(\{U\}) \cdot m_2(\{h_1\}) + m_1(\{U\}) \cdot m_2(\{h_2\}) + m_1(\{U\}) \cdot m_2(\{h_3\}) \\ &\quad + m_1(\{U\}) \cdot m_2(\{U\}) \\ &= 0.053 \times 0.208 + 0.053 \times 0.636 + 0.263 \times 0.104 + 0.265 \times 0.636 \\ &\quad + 0.159 \times 0.052 + 0.159 \times 0.636 + 0.524 \times 0.208 + 0.524 \times 0.104 \\ &\quad + 0.524 \times 0.052 + 0.524 \times 0.636 \\ &\approx 0.874\end{aligned}$$

于是

$$m(\{h_1\}) = \frac{1}{0.874} [m_1(\{h_1\}) \cdot m_2(\{h_1\}) + m_1(\{h_1\}) \cdot m_2(\{U\})]$$

$$\begin{aligned}
& + m_1(\{U\}) \cdot m_2(\{h_1\})] \\
& = \frac{1}{0.874} [0.053 \times 0.208 + 0.053 \times 0.636 + 0.524 \times 0.208] \\
& \approx 0.176
\end{aligned}$$

同理,我们可以计算得到:

$$m(\{h_2\}) \approx 0.299$$

$$m(\{h_3\}) \approx 0.157$$

而

$$m(U) = 1 - (0.176 + 0.299 + 0.157) = 0.368$$

$$\text{Bel}(H) = \sum_{h \in H} m(\{h\}) = 0.176 + 0.299 + 0.157 = 0.632$$

$$\text{Pl}(H) = 1$$

$$\begin{aligned}
f(H) &= \text{Bel}(H) + \frac{|H|}{|U|} [\text{Pl}(H) - \text{Bel}(H)] \\
&= 0.632 + \frac{3}{20} [1 - 0.632] \\
&= 0.687
\end{aligned}$$

$$\text{CER}(H) = \text{MD}(H, E') \times f(H) = 0.687$$

至此,我们求出了  $H$  的确定性为 0.687。

### 5. D-S 证据理论的特点

(1) D-S 证据理论满足比概率论更弱的公理系统。当  $|A| > 1$ , 且  $m(A) = 0$  时, 证据理论就退化为概率论; 当  $m(A_i) \neq 0$  ( $i = 1, 2, \dots, n$ ), 且有  $A_1 \subseteq A_2 \subseteq \dots \subseteq A_n$  时, 证据理论退化为可能性理论。

(2) 证据理论可以区分不知道和不确定的情况。

(3) 证据理论可以依靠证据的积累, 不断地缩小假设集合。例如, 证据可能与子集相关, 而不只是与单元素集合相关, 这样就能较好地逐步缩小诊断的目标。如: 心血管疾病  $\rightarrow$  心脏病  $\rightarrow$  左心室疾病  $\rightarrow$  专门诊断。

(4) 由 D-S 证据理论所计算的结果在数值上有时缺乏稳定性, 基本概率赋值函数 BPA 的一个很小的变化可能导致结果有较大的改变, 而且当支持的证据不一致时, 比如当两个信任函数的焦元不相交时, D-S 证据理论的规则就无法应用。

(5) D-S 证据理论要求辨别框中的元素满足相互排斥的条件, 这个条件在实际系统中不易得到满足, 而且基本概率赋值函数要求给的值太多, 计算比较复杂。

## 四、可能性理论

Zadeh 1965 年提出了模糊集合论, 1978 年又提出了可能性理论。

产生不确定性原因很多, 诸如随机性、模糊性、多义性、不完全性等。概率论作为理论基础较好地处理了随机性, 而处理模糊性的理论基础则是模糊集合论。在概率论中, 并不讨论如何选取概率值的问题, 与概率论一样, 可能性理论也不研究如何确定隶属函数值的问题, 仅讨论可能性理论的计算规则和推理规则。

### (一) 几个基本概念

可能性理论的基本思想是要确定诸如可能性、可能性分布、可能性分布函数、条件可能性

分布函数、边缘可能性分布函数等测度以及它们之间的关系。同时，它还要确定各种模糊命题的转换规则和不确定命题的推理规则等。下面，分别讨论这些内容。

## 1. 可能性与可能性分布

假设有命题

$P: x \text{ is an integer in the interval } [0, 8]$

此命题的意思“ $x$  是闭区间  $[0, 8]$  中的一个整数”。显然，该命题中的  $x$  不只是与这个区间中的某个整数有关，而且在区间  $[0, 8]$  中的任何一个整数都可能是  $x$  的一个值，而不在此区间的任何整数都不可能是  $x$  的值。根据这一事实，可以对命题  $P$  进行这样的解释，即“ $x$  is an integer in the interval  $[0, 8]$ ”引入了一个可能性分布  $\Pi_x$ ，它把每个整数  $n$  与它可能是  $x$  的一个值的可能性相联系，因此对命题  $P$  有

$$\begin{aligned}\text{poss}\{x=n\} &= 1 & 0 \leq n \leq 8 \\ \text{poss}\{x=n\} &= 0 & n < 0 \text{ 或 } n > 8\end{aligned}$$

其中， $\text{poss}\{x=n\}$  表示  $x$  取值为  $n$  的可能性。在此，区间  $[0, 8]$  中的任意一个整数  $n$  为  $x$  的值的可能性均为 1；而在区间  $[0, 8]$  以外的任意一个整数  $n$  为  $x$  的值的可能性均为 0。

下面给出更一般的可能性公理。

**可能性公理：**如果  $F$  是  $U$  的一个模糊子集，那么命题“ $x$  is  $F$ ”可以导出一个等于  $F$  的可能性分布  $\Pi_x$ ，也就是说“ $x$  is  $F$ ”可以变换成一个可能性赋值方程  $\Pi_x = F$ ，即

$$x \text{ is } F \rightarrow \Pi_x = F$$

它表示根据与  $F$  相等的可能性分布，用来定义在  $U$  中  $x$  取任何可能值的可能性。

下面再考虑一个模糊命题

$q: x \text{ is a small integer}$

此命题的意思是“ $x$  是一个小整数”，而小整数是个模糊集合，它可定义为

$$\text{small integer} = 1/0 + 1/1 + 0.8/2 + 0.6/3 + 0.4/4 + 0.2/5 + 0/6$$

其中： $y/x$  表示整数  $x$  在模糊子集中的隶属度为  $y$ ，比如  $0.8/2$  表示整数 2 在模糊子集 small integer 中的隶属度为 0.8；符号“+”表示并的关系。

命题  $q$  引入了一个可能性分布  $\Pi_x$ ，它使  $x$  取一个值  $n$  的可能性等于在模糊子集 small integer 中  $n$  的隶属度。

$$\begin{aligned}\text{poss}\{x=0\} &= 1 \\ \text{poss}\{x=1\} &= 1 \\ \text{poss}\{x=2\} &= 0.8 \\ \text{poss}\{x=3\} &= 0.6 \\ \text{poss}\{x=4\} &= 0.4 \\ \text{poss}\{x=5\} &= 0.2 \\ \text{poss}\{x=6\} &= 0\end{aligned}$$

从上述例子可以看出， $x$  的可能性分布实质上就是一个模糊子集，它给出了在  $U$  中  $x$  取任何定值的可能性。因为可能性分布的概念与模糊集合的表现形式是一致的，所以使得我们可以用模糊集合的一些运算规则对可能性分布进行操作。

## 2. 可能性分布函数

设有可能性分布  $\Pi_x = F$ ，用来定义在  $U$  中  $x$  取任何可能值的可能性。如果  $u \in U$ ，且  $u_F: U \rightarrow [0, 1]$  是  $F$  的隶属函数，那么给定“ $x$  is  $F$ ”， $x=u$  的可能性就是

$$\pi(u) \triangleq \text{poss}\{x = n \mid x \text{ is } F\} = u_F(u), u \in U$$

称  $\pi(u)$  为可能性分布函数。

### 3. 可能性赋值方程

一个自然语言的命题可以解释成一个模糊集合对一个可能性分布的赋值。或者说，如果  $P$  是一个自然语言的命题，那么  $P$  可翻译成一个可能性赋值方程：

$$P \rightarrow \Pi(x_1, x_2, \dots, x_n) = F$$

其中  $x_1, x_2, \dots, x_n$  是  $P$  中明确包含或隐含的变量。 $\Pi(x_1, x_2, \dots, x_n)$  是  $n$  个变量  $x \triangleq (x_1, x_2, \dots, x_n)$  的可能性分布。 $F$  是  $U_1 \times U_2 \times \dots \times U_n$  上的一个模糊子集，其中  $U_i (i=1, 2, \dots, n)$  是与  $x_i$  相关联的论域。

通常，一个形式为  $P \triangleq x \text{ is } F$  的命题，往往被翻译成如下形式

$$P \rightarrow \Pi_{A(x)} = F$$

其中  $A(x)$  是一个  $x$  隐藏的属性。比如

$$\text{Joe is young} \rightarrow \Pi_{\text{Age}(\text{Joe})} = \text{young}$$

$$\text{Maria is blond} \rightarrow \Pi_{\text{color}(\text{hair}(\text{Maria}))} = \text{blond}$$

$$\text{Max is about as tall as Jim} \rightarrow \Pi_{(\text{Height}(\text{Max}), \text{Height}(\text{Jim}))}$$

句子“Joe 是年轻人”、“Maria 的头发是棕色”和“Max 几乎与 Jim 一样高”中的变量隐藏的属性分别是年龄(age)、头发的颜色(hair,color)和身高(height)。

### 4. 联合可能性和条件可能性

与概率论相类似，也可以定义联合可能性和条件可能性。设  $Y_1$  和  $Y_2$  分别是  $X_1$  和  $X_2$  中取值的变量，由它们各自的分布函数，联合可能性分布和条件可能性分布定义为

$$\Pi_{(Y_1, Y_2)(x_1, x_2)} = \text{poss}\{Y_1 = x_1, Y_2 = x_2\} \quad x_1 \in X_1, x_2 \in X_2$$

和

$$\Pi_{(Y_1|Y_2)(x_1|x_2)} = \text{poss}\{Y_1 = x_1 \mid Y_2 = x_2\}$$

上面第二个式子表示当给定  $Y_2$  后， $Y_1$  的条件分布函数。

如果已知  $Y_1$  的分布函数和给定  $Y_1$  后  $Y_2$  的条件分布函数，那么我们可以由这两个函数的合取来构成  $Y_1$  和  $Y_2$  的联合分布函数：

$$\Pi_{(Y_1, Y_2)(x_1, x_2)} = \Pi_{Y_1(x_1)} \wedge \Pi_{(Y_1|Y_2)(x_2|x_1)}$$

### 5. 边缘可能性分布

在可能性理论中，边缘可能性分布是与投影操作密切相关的概念。

设  $\Pi_{(x_1, x_2, \dots, x_n)}$  是一个  $n$  元可能性分布， $s \triangleq (i_1, i_2, \dots, i_k)$  是下标序列  $(1, 2, \dots, n)$  的一个子序列，并且设  $s'$  代表补子序列，有  $s' \triangleq (j_1, j_2, \dots, j_m)$ 。例如对于  $n=5, s=(1, 3, 4)$ ，则  $s'=(2, 5)$ 。根据这种序列，一个具有形式  $(A_{i_1}, A_{i_2}, \dots, A_{i_k})$  的  $k$  元组可以缩写成  $A(s)$ 。特别地，变量  $X_{(s)} = (x_{i_1}, x_{i_2}, \dots, x_{i_k})$  可以看成  $X \triangleq (x_1, x_2, \dots, x_n)$  的  $k$  元子变量，它的补  $X(s') = (x_{j_1}, x_{j_2}, \dots, x_{j_m})$  也是子变量。

$\Pi(x_1, x_2, \dots, x_n)$  在  $U(s) \triangleq (x_{i_1}, x_{i_2}, \dots, x_{i_k})$  上的投影是一个  $k$  元的可能性分布，它可表示成

$$\Pi_{X(s)} \triangleq \text{Proj}_{U(s)} \Pi_{(x_1, x_2, \dots, x_n)}$$

并且定义

$$\pi_{X(s)}(U_{(s)}) \triangleq \sup_{s'} \Pi(s') \pi_{X(s)}(x_{i_1}, x_{i_2}, \dots, x_{i_k})$$

其中  $\pi_{X(s)}(U_{(s)})$  是  $\Pi_{X(s)}$  的可能性分布函数。例如对于  $n=2$ ，有

$$\pi_{x_1}(u_1) \triangleq \sup_{U_2} \Pi(x_1, x_2, \dots, x_n)(u_1, u_2)$$

是  $\Pi(x_1, x_2, \dots, x_n)$  在  $U_1$  上的投影的可能性分布函数的表达式。由于它与一个边缘概率分布的概率类似, 所以我们将它定义为边缘可能性分布。

边缘可能性分布的概念给我们提供了处理变量  $X_{(i)}$  和可能性分布  $\Pi_{X(i)}$  之间的关系的途径。

给定变量  $X \triangleq (x_1, x_2, \dots, x_n)$  的可能性分布  $\Pi(x_1, x_2, \dots, x_n)$ , 则子变量  $X_{(i)} \triangleq (x_{i_1}, x_{i_2}, \dots, x_{i_t})$  的可能性分布  $\Pi_{X(i)}$  可以由  $\Pi(x_1, x_2, \dots, x_n)$  向  $U_{(i)}$  上的投影获得, 即

$$\Pi_{X(i)} \triangleq \text{Proj}_{U_{(i)}} \Pi(x_1, x_2, \dots, x_n)$$

例如, 假定  $n=3, U_1=U_2=U_3=a+b$ , 记为  $\{a, b\}$ , 且  $\Pi(x_1, x_2, \dots, x_n)$  表示为一个线性的表达式。

$$\Pi_{(x_1, x_2, x_3)} = 0.8aaa + 1aab + 0.6baa + 0.2bab + 0.5bbb$$

其中具有  $0.6baa$  形式的项表示为

$$\text{poss}\{x_1=b, x_2=a, x_3=a\} = 0.6$$

为了能导出  $\Pi_{(x_1, x_2)}$ , 我们仅只从上式中的每一项中的  $x_3$  用空号代替其值, 于是

$$\begin{aligned} \Pi_{(x_1, x_2)} &= 0.8aa + 1aa + 0.6ba + 0.2ba + 0.5bb \\ &= 1aa + 0.6ba + 0.5bb \end{aligned}$$

类似地

$$\Pi_{(x_1)} = 1a + 0.6b + 0.5b = 1a + 0.6b$$

(上面运算“+”为  $\max\{\cdot\}$ )

## (二) 语言变量

日常生活中, 人们描述事件时往往使用语言而不是用数字来表示变量的值和变量之间的关系。由于语言不像数字那样精确, 因而语言变量可以用来描述那些不精确的事件或现象。

语言变量的每一个值, 实际上都是一个模糊变量。比如, 变量“年龄”可以看成是数字变量  $u \in [0, 150]$ , 同时也可以将“年龄”视作是语言变量, 可以取值为年轻、很年轻、不很年轻、老、很老、不很老等。这些值可看成是论域  $U=[0, 150]$  上模糊子集的标名, 而数字变量  $u$  称为基变量。

一个语言变量的值是建立在一个或多个原始词的基础上, 加上一组修饰词和连接词而合成的语言值。比如, 语言变量“年龄”的原始词为“年轻”、“老”, 加上修饰词和连接词后可以为“不很老且不很年轻”。语言变量的原始词通常有两个, 其中一个是另一个的反义词, 比如原始词“年轻”和“老”。

## (三) 命题模糊性的描述

模糊逻辑提供了命题模糊性描述的六种途径, 即模糊谓词、模糊量词、模糊概率、模糊可能性、模糊真值和模糊修饰词。

### 1. 模糊谓词

设  $x$  是在  $U$  中取值的某一变量,  $F$  是模糊谓词, 则命题可表示为

$$x \text{ is } F$$

其中模糊谓词  $F$  可以是描述关系的词语, 比如大和小、年轻和老、冷和热、高和矮等。例如语句

“Tom is young”可表示为  $x \text{ is } F$ , 其中  $x \triangleq \text{Age}(\text{TOM})$ ,  $F \triangleq \text{young}$ 。

## 2. 模糊量词

设  $Q$  为模糊量词,  $A$  和  $B$  为模糊谓词, 则命题可表示为

$$QA'S \text{ are } B'S$$

其中模糊量词  $Q$  可以是表示不定数词的词语。比如多数、少数、几个、经常、偶而等。例如有命题“多数歌手是年轻人”。

## 3. 模糊概率

设  $\lambda$  为模糊概率, 则可以对命题用模糊概率加以限定:

$$(x \text{ is } F) \text{ is } \lambda$$

其中模糊概率  $\lambda$  可以是“或许”、“未必”等词语。例如句子“小王也许是年轻人”。

## 4. 模糊可能性

设  $\pi$  为模糊可能性, 则可以对命题用模糊可能性加以限定:

$$(x \text{ is } F) \text{ is } \pi$$

其中模糊可能性  $\pi$  可以是“非常可能”、“很不可能”等词语。例如句子“那个人很可能是领导”。

## 5. 模糊真值

设  $\tau$  为模糊真值, 同样也可以用  $\tau$  对命题加以限定:

$$(x \text{ is } F) \text{ is } \tau$$

其中模糊真值  $\tau$  可以是“有些真”、“非常假”等词语。例如句子“那张桌子非常真是红木的”。

## 6. 模糊修饰语

设  $m$  为模糊修饰语,  $X$  是变量,  $F$  为模糊谓词, 则命题可表示为

$$X \text{ is } mF$$

其中模糊修饰语  $m$  可以是“非常”、“很”、“有些”、“绝对”等副词。

## (四) 模糊命题的转换规则

### 1. 修正规则

修正规则表示为

如果  $Y \text{ is } F \rightarrow \Pi_y = F$

那么  $Y \text{ is } mF \rightarrow \Pi_y = F^+$

其中  $m$  是模糊修饰语,  $F^+$  是  $m$  对  $F$  的一个修正结果。

当  $m = \text{not}$  (“不”、“非”等), 那么有  $F^+ = \sim F$ , 即

$$u_{F^+}(x) = 1 - u_F(x) \quad x \in [0, 1]$$

当  $m = \text{Very}$  (“很”、“非常”等), 那么有  $F^+ = F^2$ , 即

$$u_{F^+}(x) = u_F^2(x) \quad x \in [0, 1]$$

当  $m = \text{more or less}$  (“有些”、“稍微”等), 那么有  $F^+ = \sqrt{F}$ , 即

$$u_{F^+}(x) = \sqrt{u_F(x)} \quad x \in [0, 1]$$

### 2. 合取、析取和蕴含规则

设有命题

$$Y \text{ is } F \rightarrow \Pi_Y = F$$

$$Z \text{ is } G \rightarrow \Pi_Z = G$$

其中  $F$  和  $G$  分别是模糊变量  $x_1$  和  $x_2$  的模糊子集。那么两命题的合取

$$(Y \text{ is } F) \wedge (Z \text{ is } G) \rightarrow \Pi_{(Y,Z)} = F \times G$$

其中

$$u_{F \times G}(x_1, x_2) \triangleq u_F(x_1) \wedge u_G(x_2)$$

而两命题的析取

$$(Y \text{ is } F) \vee (Z \text{ is } G) \rightarrow \Pi_{(Y,Z)} = F^* \cup G^*$$

其中

$$\begin{aligned} F^* &\triangleq F \times x_2, \quad G^* \triangleq x_1 \times G \\ u_{F^* \cup G^*}(x_1, x_2) &= u_F(x_1) \vee u_G(x_2) \end{aligned}$$

进而两命题的蕴含

$$(Y \text{ is } F) \rightarrow (Z \text{ is } G) \rightarrow \Pi_{(Z|Y)} = F^* \oplus G^*$$

其中  $\Pi_{(Z|Y)}$  表示给定  $Y$  后,  $Z$  的条件可能性分布;  $\oplus$  表示有界和, 且定义为

$$u_{F^* \oplus G^*}(x_1, x_2) = 1 \wedge (1 - u_F(x_1) + u_G(x_2))$$

### 3. 量化规则

如果  $X = \{x_1, x_2, \dots, x_n\}$ ,  $Q$  是一个模糊量词, 且

$$Y \text{ is } F \rightarrow \Pi_y = F$$

那么命题“ $QY$  are  $F$ ”可以转换成如下形式

$$\Pi_{\text{count}(F)} = Q$$

其中

$$\text{count}(F) = \sum_{i=1}^n x_i$$

### 4. 真值量化规则

设  $T$  是一个模糊真值, 一个真值量化命题可以表示为“ $Y$  is  $F$  is  $T$ ”, 这类命题的转换规则可以表示成如下的形式

$$Y \text{ is } F \text{ is } T \rightarrow \Pi_y = F^+$$

其中

$$u_{F^+}(x) = u_T(u_F(x))$$

## (五) 模糊推理规则

下面讨论专家系统中常用的几种模糊推理规则。

### 1. 广义假言推理

广义假言推理是 Zadeh 提出的一种模糊推理形式, 它可以叙述为  
考虑两个命题  $\{P_1, P_2\}$ :

$$\begin{aligned} P_1 &\triangleq \text{IF } x \text{ is } F \text{ THEN } Y \text{ is } G \\ P_2 &\triangleq x \text{ is } F^* \end{aligned}$$

其中  $F, F^*$  和  $G$  为模糊谓词, 两个命题又可以写成

$$P_1 \rightarrow \Pi_{(Y|X)} = H$$

$$P_2 \rightarrow \Pi_X = F^*$$

其中  $u_H(u, v) = 1 \wedge (1 - u_F(u) + u_G(v))$

则  $Y$  的可能性分布  $\Pi_y$  为  $H$  和  $F^*$  对于  $X$  的合成:

$$\Pi_y = H \circ F^*$$

其中符号“。”表示合成运算，且有

$$\begin{aligned} u_Y(v) &= \sup_u (u_H(u, v) \wedge u_{F^*}(u)) \\ &= \sup_u (u_{F^*}(u) \wedge (1 - u_F(u) + u_F(v))) \end{aligned}$$

其中符号“ $\wedge$ ”表示  $\max, \sup$  为上确界。这个结果用三段论的形式可以表示成

$$\begin{array}{c} \text{IF } x \text{ is } F \text{ THEN } Y \text{ is } G \\ X \text{ is } F^* \\ \hline \end{array}$$

$$Y \text{ is } H \circ F^*$$

广义假言推理是经典的假言推理的一种推广。在广义假言推理中， $F^*$  和  $F$  可以是不同的， $F$ 、 $F^*$  和  $G$  作为模糊谓词，可以表达模糊的概念。当  $F=F^*$ ，且  $F$  和  $G$  为普通谓词时，广义假言推理就简化为假言推理，即

$$\begin{array}{c} \text{IF } x \text{ is } F \text{ THEN } Y \text{ is } G \\ X \text{ is } F \\ \hline \end{array}$$

$$Y \text{ is } G$$

## 2. 采用模糊量词的近似推理

不失一般性，考虑下面形式的命题：

$$P \trianglelefteq Q A' S \text{ are } B' S$$

其中  $A$  和  $B$  为模糊谓词， $Q$  为模糊量词。为方便起见， $A$  和  $B$  可以看成是  $P$  的前件和后件，而  $P$  可以表示为条件可能性分布：

$$\text{Prob}\{B|A\} \text{ is } Q$$

其中  $\text{Prob}\{B|A\}$  表示给定模糊事件  $\{x \text{ is } A\}$  时，模糊事件  $\{x \text{ is } B\}$  的条件可能性， $Q$  起到了一种模糊度量的作用。

### (1) 交/积三段论法

设  $A, B$  和  $C$  为模糊谓词， $Q_1 \otimes Q_2$  为模糊量词  $Q_1$  和  $Q_2$  的模糊积，则交/积三段论法可表示为

$$\begin{array}{c} Q_1 A' S \text{ are } B' S \\ Q_2 (A \text{ and } B)' S \text{ are } C' S \\ \hline \end{array}$$

$$(Q_1 \otimes Q_2) A' S \text{ are } (B \text{ and } C)' S$$

例如

$$\begin{array}{c} \text{多数战士是年轻人} \\ \text{多数年轻战士是男性} \\ \hline \end{array}$$

$$\text{多数}^2 \text{ 战士是年轻男性}$$

### (2) 乘积链规则

设有模糊谓词  $A$  和  $B$ ，且满足  $B \subset A$ ，而模糊量词  $Q_1$  和  $Q_2$  均为单调增加，则乘积链规则可表示为

$$\begin{array}{c} Q_1 A' S \text{ are } B' S \\ Q_2 B' S \text{ are } C' S \\ \hline \end{array}$$

$$(Q_1 \otimes Q_2) A' S \text{ are } C' S$$

例如

多数学生是本科生  
多数本科生是学工的

多数<sup>2</sup>学生是学工的

### 3. 采用模糊真值限定的近似推理

模糊真值限定的近似推理方法是 Baldwin 基于 Zadeh 的真值函数修正(Truth Functional Modification, TFM)和真值函数修正的逆(Inverse Truth Functional Modification, ITFM)提出的一种近似推理方法。

设有命题  $(X \text{ is } F)\text{is } \tau, F \subseteq U$ , 那么其等价命题是

$$X \text{ is } F', F' \subseteq U$$

其中

$$u_{F'}(u) = u_\tau(u_F(u))$$

式中的  $u_F$  和  $u_\tau$  分别为描述模糊谓词  $F$  和真值约束  $\tau$  的隶属函数, 我们称  $F'$  为给定  $\tau$  时  $F$  的真值函数修正, 并记为  $F' = \text{TFM}(F|\tau)$ 。

设有命题  $X \text{ is } F, F \subseteq U$ , 且知道  $X \text{ is } F', F' \subseteq U$ , 那么给定  $X \text{ is } F'$  时命题  $X \text{ is } F$  的真值约束为

$$u_\tau(\eta) = \sup_{\substack{u \in U \\ u_F(u) = \eta}} u_F(u)$$

这时命题  $X \text{ is } F$  修正为  $(X \text{ is } F)\text{is } \tau$ , 并记为  $\tau = \text{ITM}(F|F')$ 。

由上述的定义, Baldwin 导出的近似推理形式为

$$\begin{array}{c} (\text{IF } X \text{ is } F \text{ THEN } Y \text{ is } G)\text{is } \tau & F \subseteq U, G \subseteq V \\ X \text{ is } F' & F' \subseteq U \\ \hline Y \text{ is } G' & G' \subseteq V \end{array}$$

其中

$$\begin{aligned} G' &= \text{TFM}(G|\text{ITFM}\cdot(F|F') \circ I(\tau)) \\ \mu_{I(\tau)}(u, v) &= \mu_\tau((1-u+v) \wedge 1) \quad \forall u \in U, \forall v \in V \end{aligned}$$

## (六) 可能性理论的优缺点

1. 对由于词语不精确而产生的一些不确定性问题, 可能性理论是一个较好的方法。
2. 可能性理论提供了多种运算定义和多种问题形式化的方法。
3. 可能性理论具有较低的信息和时间复杂度。
4. 可能性理论所使用的隶属函数构造缺乏相应标准, 其值带有强烈的主观因素。
5. 可能性理论要求针对不同情况相应地选择不同的运算定义, 这对实际应用造成一定困难。

## 五、信念网络

信念网络(Belief Network)亦称为贝叶斯网络, 由 Judea Pearl 1985年提出。它是一种模拟人类推理过程中因果关系的不确定性处理模型。网络的拓扑结构是一个无圈的有向图, 网络中

的结点表示问题求解中的命题或变量，也可以是假设，网络通过弧把认为有直接关系的命题或变量连接起来。如果网络中结点  $M$  的值直接影响到结点  $N$  的值（记作  $M \rightarrow N$ ），那么就可以用弧把结点  $M$  和  $N$  连接起来，且连接强度用条件概率  $P(N|M)$  来表示。

一般来说，具有  $n$  个命题  $x_1, x_2, \dots, x_n$  的系统，各命题之间的相互关系所呈现出的知识可用联合概率分布函数来描述。然而，这样处理使得问题过于复杂，Pearl 认为人类在推理过程中，知识并不是以联合概率分布形式表现，而是以变量之间的相关性和条件相关性表现的，这可以用条件概率来表示。如：

$$P(x_1, x_2, \dots, x_n) = P(x_n | x_{n-1}, \dots, x_1) \cdot P(x_{n-1} | x_{n-2}, \dots, x_1) \cdots \\ \cdot P(x_2 | x_1) \cdot P(x_1)$$

对如图 6-6 所示的信念网络，有

$$P(x_1, x_2, x_3, x_4, x_5, x_6) = P(x_6 | x_5) \cdot P(x_5 | x_2, x_3) \cdot P(x_4 | x_1, x_2) \cdot \\ P(x_3 | x_1) \cdot P(x_2 | x_1) \cdot P(x_1)$$

一旦命题之间的相关性由边表示，条件概率由边的强度来表示，则命题之间静态结构关系的有关知识就表示出来了。

当系统获取某个新的证据事实时，要对每个命题的可能取值加以综合考查，进而对每个结点定义一个信任度，记作  $\text{Bel}(x)$ ，且规定

$$\text{Bel}(x) = P(x = x_i | D)$$

来表示当前所具有的所有事实和证据  $D$  条件下，命题  $x$  取值为  $x_i$  的可信任程度，然后再基于  $\text{Bel}$  计算新的证据和事实下各命题的可信任程度。

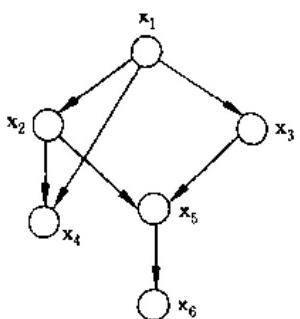


图 6-6 信念网络

### 第三节 非单调推理

建立在谓词逻辑基础上的传统逻辑是单调的，其单调含义指已知为真的命题数目随着推理的进行而严格地增加。在单调逻辑中，新的命题可以加入系统，新的定理可以被证明，并且这种加入和证明绝不会导致前面已知的命题或已证明的命题变成无效。

遗憾的是，人类的思维及推理活动在本质上并非单调的。人们对周围世界中的事物的认识、信念和观点，总是处于不断地调整之中。比如，根据某些前提推出某一结论，但当人们又获得另外一些事实后，却又取消这一结论。在这种情况下，结论并不随着条件的增加而增加，这种推理过程就是非单调推理。

例如，公安人员进行刑事侦察时的推理就是非单调的。某单位金库被盗，公安人员根据对现场的调查，推测作案人是  $A$ ，可后来得知失窃的那天  $A$  恰好在  $C$  市出差，于是排除了  $A$  作案的可能。但后来又进一步了解到在失窃的那天晚上  $A$  并没有在  $C$  市客店中，而是骑着摩托车出了客店，则又再一次推测  $A$  是连夜返回本市作案。最后在  $A$  的家中取得了失窃的现款，才算完成了全部推理，而知道  $A$  到  $C$  市出差只不过是掩人耳目。

传统逻辑是不能进行上述非单调推理的。因为传统逻辑在某些前提下得出某一结论  $P$  后，就不能再改变了。如果加入新的事实  $Q$  后能导出非  $P$ ，则新的系统是矛盾的，这种情况在传统逻辑中是不允许的。

非单调逻辑已成为人工智能研究中非常活跃的领域。目前,在非单调推理研究中代表性的工作有:R·Reiter 的缺省推理(缺省逻辑)、Moore 的自认识逻辑、J·McCarthy 的界限推理和 Doyle 的正确性维护系统。其中,缺省推理是指在  $S$  缺省条件下成立,就意味着“当且仅当没有事实证明  $S$  不成立时, $S$  是成立的”。自认识逻辑的基本思想是:如果知道  $S$ ,并且不知道有其他任何事实与  $S$  矛盾,那么  $S$  是成立的”。界限推理的原则是“当且仅当没有事实证明  $S$  在更大的范围内成立时, $S$  只有指定的范围中成立”。而正确性维护系统是运用非单调推理的思想来维护知识库,就得到正确性维护系统。在这种系统中,每个知识单元都是信念,每个信念都有其正面或反面的论据,在推理过程中论据发生了变化,信念也随之而发生变化。

实现非单调推理方法有两种,一种是在经典逻辑中增加某些公理,用以导出非单调推理的结果(如限定推理);另一种是定义特定的非经典逻辑(如缺省理论和自认识逻辑)。这两种方法各有千秋。

## 一、缺省推理

人工智能中所涉及到的缺省推理问题可叙述为:在没有证据可以证明某事件不存在的情况下,就承认该事件的存在。这也就是说尽管不具备事件的全部知识,也能作出合理的推理和结论。

例如,通常认为,高等学校的教师都会英语。当谈到某一位教师李华时,在没有任何信息可以证明他不懂英语的情况下,根据一般常识认为他会英语,因此得出他懂英语这一结论。如果后来得到进一步的事实,证明他并不懂英语时,可对这个结论进行修正。对于这类问题可以使用缺省推理规则处理。该规则可描述为:“当  $x$  是高等学校的教师,如果没有相反的证据,那么应认为  $x$  掌握英语。”规则可形式化为

$$\frac{\text{Teacher}(x), M \text{Master-English}(x)}{\text{Master-English}(x)}$$

其中  $M$  为模态算子,表示“如果假设…没有矛盾”或者表示“假设……相容”。这样,上式就可表示为:若  $x$  是教师,而且假定他掌握英语没有问题,那么,可以认为他掌握英语。显然,引入缺省推理后,可以解决一些模糊量词的表示问题,比如“大多数”、“通常是”、“几乎都是”等。

### (一) 缺省推理规则的表示

缺省推理规则的一般形式可表示为

$$\frac{\alpha(x), M\beta_1(x), \dots, M\beta_m(x)}{\omega(x)}$$

其中  $\alpha(x), \beta_1(x), \dots, \beta_m(x), \omega(x)$  是公式。 $\alpha(x)$  称为规则的前提, $\omega(x)$  称为规则的结论, $\beta_i(x)$  ( $i=1, 2, \dots, m$ ) 称为缺省条件。

一个缺省理论可以表示成一个二元组:

$$\Delta = (D, W)$$

其中  $D$  是缺省规则集, $W$  是合式公式集合。

对于缺省理论  $\Delta = (D, W)$  和任意公式集  $S \subseteq L$ ,令  $\Gamma(S)$  是满足下面条件的最小集合:

- (1)  $W \subseteq \Gamma(S)$
- (2)  $Th_L(\Gamma(S)) = \Gamma(S)$

(3) 如果  $(\alpha; M\beta_1, \dots, M\beta_m / \omega) \in D$ ,  $\alpha \in \Gamma(S)$ , 并且  $\sim\beta_1, \dots, \sim\beta_m \notin S$ , 那么  $w \in \Gamma(S)$   
其中  $Th_L(S)$  是由  $S$  生成的理论。

于是, 算子  $\Gamma$  的一个不动点  $E$  为缺省理论  $\Delta$  的一个信念延拓, 即  $\Gamma(E) = E$ 。信念延拓  $E$  是关于世界  $W$  的可接受的信念集合, 它包括初始假设  $W$  以及应用缺省规则推出的缺省假设。

## (二) 缺省规则的分类

缺省规则按其规范性可分类为:

### 1. 规范缺省理论

将形如  $\frac{\alpha(x); M\omega(x)}{\omega(x)}$  的缺省规则称为规范的。其中  $\alpha(x), \omega(x)$  是任意公式。缺省理论  $(D, W)$  称为规范的缺省理论, 如果  $D$  中每一缺省规则都是规范的。规范的缺省理论有两种不同的情况, 它们是:

#### (1) 形式为

$$\frac{\alpha(x); M\beta(x)}{\beta(x)}$$

其含义是: 由  $\alpha(x)$  一般可以推出  $\beta(x)$ 。例如有命题“大多数鸟都会飞”可以表示成

$$\frac{\text{Bird}(x); M\text{Can-fly}(x)}{\text{Can-fly}(x)}$$

#### (2) 形式为

$$\frac{\alpha(x); M\neg\beta(x)}{\neg\beta(x)}$$

其含义是: 由  $\alpha(x)$  一般可以推出  $\neg\beta(x)$ 。该形式适用于封闭世界假设 (Closed World Assumption, CWA)。所谓封闭世界假设是在已有信任集  $\Delta$  下实现完备化的最直接方法。

CWA 方法是在已有的信任集  $\Delta$  的情况下, 再考虑一个假设集  $\Delta_{\text{asm}}$ , 使得  $\Delta \cup \Delta_{\text{asm}}$  完备。

设  $T[\Delta]$  是由  $\Delta$  形成的理论集,  $\varphi \in T[\Delta]$ , 当且仅当  $\Delta \vdash \varphi$ 。而  $CWA[\Delta]$  是由  $\Delta \cup \Delta_{\text{asm}}$  形成的理论集,  $\varphi \in CWA[\Delta]$ , 当且仅当  $\Delta \cup \Delta_{\text{asm}} \vdash \varphi$ 。

CWA 方法是这样来定义  $\Delta_{\text{asm}}$  的。如果  $\neg P \in \Delta_{\text{asm}}$  当且仅当  $P \notin T[\Delta]$ , 即在  $\Delta$  下,  $P$  是不可证明的, 就假设  $\neg P$  是可以证明的, 这样的  $\neg P$  就放入  $\Delta_{\text{asm}}$  来扩充  $T[\Delta]$ 。

CWA 广泛应用于数据库系统和知识库系统。例如, 当联机检索一篇文献时, 如果系统没有录入该文献, 系统将提示该文献不存在。

### 2. 半规范缺省理论

半规范缺省理论可以有下面的两种形式:

#### (1) 形式为

$$\frac{\alpha(x); M(\beta(x) \wedge \neg\gamma(x))}{\beta(x)}$$

其中  $\beta(x) = \omega(x) \wedge \gamma(x)$ 。

这种形式的缺省理论含义是“除非  $\gamma(x)$ , 由  $\alpha(x)$  一般可以推出  $\beta(x)$ ”。例如, 命题“除了企鹅之外, 大多数的鸟都会飞。”可以描述为:

$$\frac{\text{Bird}(x); M(\text{Can-fly}(x) \wedge \neg\text{Penguin}(x))}{\text{Can-fly}(x)}$$

#### (2) 形式为

$$\frac{\alpha(x); M(\sim\beta(x) \wedge \sim\gamma(x))}{\sim\beta(x)}$$

其中  $\beta(x) = \omega(x) \wedge \gamma(x)$ 。

这种形式的缺省理论含义是“除非  $\gamma(x)$ , 由  $\alpha(x)$  一般可以推出  $\sim\beta(x)$ ”。例如, 命题“除了 Tom 之外, 大多数美国人不会汉语。”可以描述为:

$$\frac{\text{American}(x); M(\sim\text{Master-Chinese}(x) \wedge \sim\text{Tom}(x))}{\sim\text{Master-Chinese}(x)}$$

## 二、自认识逻辑

### (一) 自认识逻辑的描述

设  $A$  是一个命题,  $\underline{L}$  和  $\underline{M}$  分别表示“相信”和“可接受”算子, 那么  $\underline{L}A$  表示相信命题  $A$ ,  $\underline{M}A$  表示可接受命题  $A$ , 且它们之间的关系是:

$$\underline{M}A = \sim\underline{L}\sim A$$

此式可解释为: 如果推理者并不相信  $A$  的非, 那么  $A$  和推理者的当前知识是一致的。

自认识逻辑系统 AE 的合式公式定义为:

1. 传统命题逻辑的合式公式都是 AE 系统的合式公式;
2. 若  $A$  是合式公式, 则  $\underline{L}A$  和  $\underline{M}A$  也是合式公式;
3. 若  $A, B$  是合式公式, 则  $\sim A, A \vee B, A \wedge B, A \rightarrow B$  和  $A \equiv B$  也是合式公式。

自认识逻辑系统 AE 中的命题按下面的规则确定其真假值:

1. 不含算子  $\underline{L}$  和  $\underline{M}$  的命题按传统命题逻辑指派真假值;
2. 若  $A, B$  是真假值已知的合式公式, 则  $\sim A, A \vee B, A \wedge B, A \rightarrow B$  和  $A \equiv B$  的真假值按传统逻辑中对  $\sim, \vee, \wedge, \rightarrow, \equiv$  这五种联结词的语义进行赋值;
3. 传统命题逻辑中的推理规则适用于 AE。

若自认识逻辑系统 AE 中的命题真值指派符合下面两条规则, 那么 AE 称为是稳定的自认识逻辑系统:

1. 若命题  $A$  为真, 则命题  $\underline{L}A$  亦为真。
2. 若命题  $A$  为假, 则命题  $\underline{L}A$  亦为假。

### (二) 自认识逻辑系统 AE 的语义

一个具体 AE 系统全部命题的集合称为一个自认识理论, 其中包括了一个推理者所有的全部信念的集合。

可对一个自认识理论中的全部命题进行真值指派, 该真值指派符合传统逻辑中所有真值指派规则。对  $\underline{L}A$  形式命题的真值指派除不违反上述规定外可以是任意的, 这种真值的指派称为自认识理论的一个解释。

设  $I$  是自认识理论的一个解释, 那么解释中全体被指派为真的命题构成自认识理论的一个模型。

自认识理论的一个解释  $I$  称为是该理论的一个自认识解释。如果命题  $A$  被指派为真, 当且仅当命题  $\underline{L}A$  被指派为真。

设  $I$  是自认识理论的一个自认识解释,那么解释中全体被指派为真的命题构成自认识理论的一个自认识模型。

一个自认识理论  $T$  称为是相对于公理集  $A$  健康的,当且仅当如果  $T$  的一个自认识解释是公理集  $A$  在传统命题逻辑意义下的模型,那么它是  $T$  的一个模型。

一个自认识理论  $T$  称为是语义完备的,当且仅当如果  $A$  是一个在  $T$  的所有自认识模型中均真的命题,那么  $A$  本身包含在  $T$  中。

稳定的自认识理论被 Moore 比喻为一个理想的推理者,他在已有公理的前提下无所不知,而且所知必真。

一个自认识理论  $T$  是相对于公理集  $A$  健康的,当且仅当  $T$  是忠实于  $A$  的。一个稳定的且忠实于公理集  $A$  的自认识理论  $T$  称为是  $A$  的一个稳定的扩张。

### 三、界限理论

界限理论是最早研究非单调推理的方法之一,其核心思想是所谓的“Occam 刃”原理,即如果一个句子叙述命题,那么它叙述的仅仅是这个命题,一点都不能扩张和伸展。任何多余的东西都要用这把“Occam 刃”剥掉。例如我们说笔可以写字,那就意味着只有笔才能写字,其他任何可以写字的工具都要被刃剥去。

McCarthy 把 Occam 刃称为最小模型。

#### (一) 最小模型

McCarthy 的最小模型定义为:

对已知的信任集  $\Delta$ ,有  $M$  和  $M^*$  是  $\Delta$  的两个模型(使得  $\Delta$  成立的解释),满足

(1)  $M$  和  $M^*$  有相同的个体域

(2)  $M$  和  $M^*$  对  $\Delta$  中除  $P$  以外的所有其他谓词和常项有相同的设定

(3)  $M^*$  中满足  $P$  的个体集是  $M$  中满足  $P$  的个体集的子集

便有

$$M^* \leqslant_P M$$

且称  $M^*$  是  $M$  的一个子模型。如果上面条件能使得

$$M^* \leqslant_P M$$

则称  $M^*$  是  $M$  的一个真子模型。

没有真子模型的模型称为最小模型。

例如 设论域  $D=\{1,2\}$  上有

$$\Delta = (\forall x)(\exists y)(P(y) \wedge Q(x,y))$$

$$= [(P(1) \wedge Q(1,1)) \vee ((P(2) \wedge Q(1,2))] \wedge [(P(1) \wedge Q(2,1)) \\ \vee ((P(2) \wedge Q(2,2))]$$

若模型

| $M; P(1)$   | $P(2)$ | $Q(1,1)$ | $Q(1,2)$ | $Q(2,1)$ | $Q(2,2)$ |
|-------------|--------|----------|----------|----------|----------|
| $T$         | $T$    | $F$      | $T$      | $F$      | $T$      |
| $M^*; P(1)$ | $P(2)$ | $Q(1,1)$ | $Q(1,2)$ | $Q(2,1)$ | $Q(2,2)$ |
| $F$         | $T$    | $F$      | $T$      | $F$      | $T$      |

显然,  $\Delta|_M = T$ ,  $\Delta|_{M^*} = T$ , 所以  $M, M^*$  都是  $\Delta$  的模型。在这两个模型  $M$  和  $M^*$  中, 对于  $Q$  的真值设定是相同的, 而对于  $P$  来说就不相同了。 $M$  中有  $\{1, 2\}$  中的元素使得  $P$  为真, 而  $M^*$  中仅有  $\{2\}$  中的 2 使得  $P$  为真, 即  $\{2\} \subseteq \{1, 2\}$ 。

于是我们有

$$M^* \leqslant_p M$$

对于  $\Delta$  来说, 如果对任一  $M \leqslant_p M_n$ , 那么必有  $M = M_n$  成立时, 就称  $M_n$  是  $P$  的最小模型。需要指出的是, 对任一个  $\Delta$  来说, 最小模型不一定总是存在的。

## (二) 按最小化原则建立假设

给定含有谓词  $P$  的信任集  $\Delta$ , 寻找对  $\Delta$  的假设扩充公式  $\Phi_P$ , 使得对于  $\Delta \wedge \Phi_P$  的任一模型  $M$ , 不存在  $\Delta$  的模型  $M^*$ , 满足

$$M^* <_p M$$

或者说扩充后的  $\Delta \wedge \Phi_P$  的模型  $M$  对  $P$  来说不能比原来  $\Delta$  的模型  $M^*$  来得大, 这是一种最小的扩充。按这种最小化原则建立的  $\Delta \wedge \Phi_P$ , 是  $\Delta$  对  $P$  的一种界限。

McCarthy 提出了界限推理的形式化方法。比如下面有关谓词的界限。

设  $P^*$  是某个谓词常项, 它与  $P$  有相同的变元个数, 由  $P^*$  来构造  $\Phi_P$ , 可以证明

$$(\forall x)(P^*(x) \rightarrow P(x)) \wedge \sim(\forall x)(P(x) \rightarrow P^*(x)) \wedge \Delta(P^*)$$

的任何模型都不是  $\Delta$  对  $P$  的最小模型。而

$$\sim((\forall x)(P^*(x) \rightarrow P(x)) \wedge \sim(\forall x)(P(x) \rightarrow P^*(x)) \wedge \Delta(P^*))$$

的任一模型是  $\Delta$  对  $P$  的最小模型。于是

$$\Phi_P = (\forall P^*) \sim((\forall x)(P^*(x) \rightarrow P(x)) \wedge \sim(\forall x)(P(x) \rightarrow P^*(x)) \wedge \Delta(P^*))$$

而

$$\begin{aligned} \Delta \wedge \Phi_P &= \Delta \wedge ((\forall P^*) \sim((\forall x)(P^*(x) \rightarrow P(x)) \wedge \sim(\forall x)(P(x) \rightarrow \\ &\quad \rightarrow P^*(x)) \wedge \Delta(P^*))) \\ &= \Delta \wedge (\forall P^*) ((\Delta(P^*) \wedge (\forall x)(P^*(x) \rightarrow P(x))) \rightarrow (\forall x)(P(x) \rightarrow P^*(x))) \end{aligned}$$

即

$$\Phi_P = (\forall P^*) ((\Delta(P^*) \wedge (\forall x)(P^*(x) \rightarrow P(x))) \rightarrow (\forall x)(P(x) \rightarrow P^*(x)))$$

这个结果说明: 如果求得  $P^*$  使  $\Delta(P^*)$  成立, 又满足  $(\forall x)(P^*(x) \rightarrow P(x))$ , 那么  $(\forall x)(P(x) \rightarrow P^*(x))$  就是推理出的结论。

倒 以积木世界为例, 说明上述形式化的意义。

设  $\Delta = \text{is Block}(a) \wedge \text{is Block}(b) \wedge \text{is Block}(c)$

这里谓词  $P(x) = \text{is Block}$  限制在积木块  $a, b$  和  $c$  范围。 $\Delta$  可写成  $\Delta(P)$ 。于是

$$\Phi_P = \Delta(P^*) \wedge (\forall x)(P^*(x) \rightarrow \text{is Block}(x)) \rightarrow (\forall x)(\text{is Block}(x) \rightarrow P^*(x))$$

选取

$$\begin{aligned} P^*(x) &= (x=a) \vee (x=b) \vee (x=c) \\ \Delta(P^*) &= ((a=a) \vee (a=b) \vee (a=c)) \wedge \\ &\quad ((b=b) \vee (b=c)) \wedge \\ &\quad ((c=c)) \\ &= T \end{aligned}$$

而

$$(\forall x)(P^+(x) \rightarrow \text{is Block}(x)) = T$$

于是,我们有

$$(\forall x)(\text{is Block}(x) \rightarrow ((x=a) \vee (x=b) \vee (x=c)))$$

这个式子表明,如果把谓词 is Block 限制在  $\Delta$  所限制的范围内,那么  $a, b, c$  就是所有的积木。这个推理显然是非单调的,因为将其他积木块加入  $\Delta$  中,该结论就不再成立了。

界限理论的推理能够使机器在有限的条件下,立即得出结论或作出决策,而不去考虑大量没有明确加入说明的条件,这些条件从常识来看都应该是成立的,是不言而喻的。事实上,这也是人类进行常识推理所运用的原则。

## 四、正确性维持系统

前面介绍的三种非单调推理理论都没有涉及对信念的修正,Doyle 的正确性维持系统 (TMS, Truth Maintenance System) 是解决信念修正问题最早的程序。该程序可以协助其他的推理方法来维持系统的正确性,它的功能并不是生成新的推理,而是在其他程序所产生的命题之间保持相容性,若发现某个命题不相容,它就通过自己的推理机制,面向从属关系进行回溯,并用修改最小信念集的方法来消除这种不相容。

在 TMS 中,每一个命题或规则均称为结点,对任一结点,下面的两种状态必居其一:

IN 相信为真

OUT 不相信为真,或无理由相信为真,或当前没有可相信的理由。

每个结点附有一张支持表,以说明该结点与其他结点的 IN 或 OUT 状态的依赖关系。支持表(SL)的形式为:

$$(SL \langle IN-\text{结点} \rangle \langle OUT-\text{结点} \rangle)$$

而条件证明的形式为

$$\begin{aligned} & (CP \quad (\text{结论})) \\ & \quad (IN-\text{假设}) \\ & \quad (OUT-\text{假设})) \end{aligned}$$

在支持表中,如果在 IN 结点表中提到的结点当前都是 IN,且在 OUT 结点表中提到的结点当前都是 OUT,那么它们是有效的。例如有三个结点:

(1)现在是冬天 支持表为(SL( ))( )

(2)天气是寒冷的 支持表为(SL(1)(3))

(3)天气是温暖的 支持表为(SL( ))( )

结点(1)和(3)的 SL 表中的 IN 和 OUT 均为空,表示它们不依赖于任何别的结点中当前的信念或缺少信念,具有这种性质的结点称为前提。而结点(2)的 SL 中的 IN 表含有结点(1),OUT 表中含有结点(3),说明这时若结点(1)是 IN 状态,结点(3)是 OUT 状态,那么结点(2)才是 IN 状态。即表示“如果现在是冬天,又没有天气是温暖的证据,则结论为天气是寒冷的。”一旦某时刻出现了天气是温暖的证据,即结点(3)是可信了,那么 TMS 将会使结点(2)转变成 OUT 状态了。

值得注意的是,TMS 本身并不产生证实,结点(2)的证实来自于冬天一般天气是寒冷的这样一个领域知识。由此,这个证实必须由 TMS 的问题求解程序提供,TMS 能做的仅是利用证实来维持一个相容的信念数据库。

又例如,找一个适当的时间使三个忙人能够同时参加会议。一个办法是首先假设一个开会的日期,比如星期三,并将关于此假设的命题放入数据库中,再从三个人的时间安排表中检查不相容性。如果出现冲突,就表示假设的命题应该取消,而用另一个希望不矛盾的命题来代替,于是又得到一个非单调系统。这个非单调系统可表示为

结点(1) 日期(会议)=星期三 (SL( ) (2))

结点(2) 日期(会议)≠星期三

目前没有相信“开会日期不应是星期三”的证实,所以结点(1)是以 IN 表示日期为星期三这一假设。经过某些推理后,安排会议系统得出会议必须在14:00举行的结论,这是根据若干结点(其编号为57、103、45)得出的。因此,有如下结点:

结点(1) 日期(会议)=星期三 (SL( ) (2))

结点(2) 日期(会议)≠星期三

结点(3) 时间(会议)=14:00 (SL(57,103,45)( ))

结点(1)、(3)为 IN, 结点(2)为 OUT, 现在要为会议找一间房间, 结果发现星期三下午两点钟无空房可供会议使用, 于是通过产生下一结点来告诉 TMS:

结点(4)矛盾 (SL(1,3)( ))

这时, 系统调用面向从属关系的回溯过程, 查看矛盾结点的 SL 证实中的结点。比如说  $A_1, \dots, A_4$ , 接着向后跟踪, 通过  $A_i$  的 SL 证实中的结点, 比如说  $B_1, \dots, B_n$ , 再回到  $B$  的 SL 证实中的结点, 继续去寻找假设, 试图找到这样的一个假设集, 只要除去该集中的一一个假设, 矛盾就可消除。

在该例中, 这个集只包含一个元素, 即结点(1), 回溯机制通过产生一个不相容结点来记录, 不相容结点表示不相容的假设集, 于是得到下面的结点集:

结点(1) 日期(会议)=星期三 (SL( ) (2))

结点(2) 日期(会议)≠星期三

结点(3) 时间(会议)=14:00 (SL(57,103,45)( ))

结点(4) 矛盾 (SL(1,3)( ))

结点(5) 不相容 N-1 (CP4(1,3)( ))

现在 TMS 选择不相容假设中的一个(此例只有结点(1)), 并通过使结点(1)的 OUT 表中的一个结点变为 IN 来使结点(1)变为 OUT。因为一切假设都有非空的 OUT 表, 所以可以这样做。该例中又只有一个结点能使其为 IN。使结点(2)为 IN 的方法是: 为结点(2)提供一个以不相容结点为根据的证实。于是有:

结点(1) 日期(会议)=星期三 (SL( ) (2))

结点(2) 日期(会议)≠星期三 (SL(5)( ))

结点(3) 时间(会议)=14:00 (SL(57,103,45)( ))

结点(4) 矛盾

结点(5) 不相容 N-1 (CP4(1,3)( ))

结点(2)和结点(5)为 IN, 就会使得结点(1)为 OUT, 结点(4)现在也变成 OUT。因此, 矛盾就消除了, 可选择一个新的日期。由于矛盾中不包含时间, 所以仍保持14:00不变。为此而要做的工作就不再重复说明。

CP 证实的重要性现在可以看到, 如果不按照上面那样证实结点(5), 而仅接受证实:

结点(5) 不相容 N-1 (SL(4)( ))

就是说它仅仅依赖于结点(4)为 IN, 它对结点(1)和(3)的依赖性将通过结点(4)对两结点的依赖性表示, 但当结点(4)变成 OUT 时, 结点(5)也要变成 OUT, 那又会引起结点(2)再变成 OUT, 且结点(1)再变成 IN。CP 证实则可避免出现这种情况, 因为它仅是记录一个逻辑推导, 而推导的成立与其中的结点当前是 IN 还是 OUT 无关。

## 第四节 粗集理论

粗集理论(Rough Set Theory, RST)是波兰华沙理工大学 Z·Pawlak 教授1982年首先提出的, 后来经过包括他本人在内的许多科学家进一步研究, 使该理论得到了发展。RST 特别适用于不要求精确数值结果的不确定性问题。该方法以对观察和测量所得数据进行分类的能力为基础, 以集合论为数学工具完成对不确定性知识的处理。

### 一、RST 的概述

**定义1** 令  $a_i(i=1,2,\dots,n)$  是 对象的一个基本概念或属性, 则  $a_i$  的有限集合记为  $A = \{a_1, a_2, \dots, a_n\}$ , 称为一个概念集。

**定义2** 由概念集  $A$  中的元素所描述的被操作的对象  $e_j(j=1,2,\dots,m)$  构成的集合, 记为  $E = \{e_1, e_2, \dots, e_m\}$ , 称为一个对象集。

例如, 描述一手非点数扑克牌的概念集为:

$A = \{\text{Spade}, \text{Head}, \text{Diamond}, \text{Club}, \text{J}, \text{Q}, \text{K}, \text{Joker}\}$ ; 而  $E = \{\text{Spade J}, \text{Spade K}, \text{Head Q}, \text{Diamond J}, \text{Diamond Q}, \text{Diamond K}, \text{Club Q}, \text{Joker}\}$  可以构成一个对象集。

如果  $e_j(j=1,2,\dots,m)$  的描述仅基于某个  $a_k(k=1,2,\dots,n)$ , 那么会产生某些  $e_j$  是不可辨别的, 因为辨别它们的属性不在概念集  $A$  中。比如, 我们不能辨别 spade 3 与 spade 5 这两张牌, 因为在概念集  $A$  中没有点数牌的描述。

**定理1** 当有限集  $A$  有定义, 就存在有一个等价关系  $\approx$ , 比如  $e_i \approx e_k$ 。这时,  $e_i$  和  $e_k$  对于所给定的  $A$  彼此是不可辨别的, 对于任意一个  $i$ ,  $a_i$  是  $e_i$  的一个属性, 当且仅当它是  $e_k$  的一个属性。

由定理1可知, 存在一个关于  $E$  的分类:

$$P = \{P_1, P_2, \dots, P_r\}$$

其中  $\bigcup P_i = E$  且  $P_i \cap P_j = \emptyset (i \neq j = 1, 2, \dots, r)$ , 称  $P_i$  是一个等价类(其中符号  $\bigcup$  和  $\cap$  分别为并和交运算)。上例中, 有关  $E$  的分类是:

$$P = \{\{\text{Spade J}, \text{Spade K}\}, \{\text{Head Q}\}, \{\text{Diamond J}, \text{Diamond Q}, \text{Diamond K}\}, \{\text{Club Q}, \text{Club K}\}, \{\text{Joker}\}\}$$

**定义3** 令  $t \in T = \{t_1, t_2, \dots, t_p\}$  是一对象,  $t$  的属性  $t_a$  有关系  $t_a \subseteq A$ , 则  $t_a$  必由属性的分类集  $P$  的项  $e_i$  来描述。于是有:

$$t^c(P, E) = \{e_i : e_i \in P_i, P_i \subseteq t\}$$

$$t^e(P, E) = \{e_i : e_i \in P_i, P_i \cap t \neq \emptyset\}$$

则称  $t^c(P, E)$  是  $E$  和  $P$  上  $t$  的核, 而  $t^e(P, E)$  是  $E$  和  $P$  上  $t$  的包络。

由定义3可知, 对于  $t$  的核,  $E$  中所有等价对象集的所有属性  $T$  都具有, 而对于  $t$  的包络,  $E$  中所有的对象集至少有一个属性为  $T$  所具有。

在上例中,若  $t = \{\text{Spade}, \text{Joker}\}$  则

$$t^c(P, E) = \{\text{Joker}\}$$

$$t^e(P, E) = \{\text{Spade J, Spade K, Joker}\}$$

定义4 由核  $t^c(P, E)$  和包络  $t^e(P, E)$  组成的对偶  $[t^c(P, E), t^e(P, E)]$  称为一个粗集。

定义5 如果一个等价对象集  $E$  中仅含有它的包络  $t^e(P, E)$ , 而不会它的核  $t^c(P, E)$ , 则称这样的集合为  $T$  的边界(boundary), 记为  $t^b(P, E)$ 。即

$$t^b(P, E) = t^e(P, E) - t^c(P, E)$$

定义6  $E$  中不包含  $t$  描述的所有等价对象的集合称为无关集, 记为  $t^u(P, E)$ 。即

$$t^u(P, E) = P - t^c(P, E)$$

令由被分类的  $P$  的  $E$  所定义的所有粗集集合为  $R$ , 有对偶  $R = [R^c, R^e]$  和  $R' = [R'^c, R'^e]$   $\in R$ 。显然, 下列集合关系式是成立的:

$$(R \cup R')^c \supseteq R^c \cup R'^c$$

$$(R \cup R')^e = R^e \cup R'^e$$

$$(R \cap R')^c = R^c \cap R'^c$$

$$(R \cap R')^e \subseteq R^e \cap R'^e$$

$$(\sim R)^c = \sim (R^c)$$

$$(\sim R)^e = \sim (R^e)$$

$$(R \rightarrow R')^c \supseteq R^c \cup R'^c$$

$$(R \rightarrow R')^e = \sim R^e \cup R'^e$$

## 二、粗集理论的不确定性知识表示

不确定性问题的一般处理方法属于简化方法, 这些方法针对事件的具体情况给出一个主观的具有一定精确度的观察值。然而, 现实世界中的事件之所以是不确定的, 表现为它们不够明显地确定是真还是假。事实上, 这些值并不是简单地依赖于问题的外部表象, 而是取决于事实之间的内部联系和事实与推理之间的关系。这就是不确定事件区别于确定事件的重要因素。在不确定性系统中, 因为事实中蕴含的关系并不总是明显地表现出来, 有时被认为相当重要的事实也不一定在与此相关的关系中显露, 这就导致了事实之间关系的不确定性。

例如, 在医疗诊断系统中, 若症候  $s_1$  不是唯一地由疾病  $D$  引起的, 那么可能说,  $s_1 \rightarrow D$  的置信度为 CF。这就是说, 疾病  $D$  的症候还可能有  $s_2, s_3, \dots$ 。在一般情况下, 我们得不到  $D$  到底是否由  $s_1$  引起的, 还是由  $s_2, s_3, \dots$  引起的确切结论, 然而, RST 可以很好地模拟这种现象。在 RST 中, 其基本依据是系统中事实所描述基本概念(属性)的集合。由前面讨论可知, 这里存在有映象, 比如粗集中的概念集与观察事实之间的映射, 概念集与系统辨别的假设集之间的映射等。

RST 应用于不确定知识处理的基本思想是将知识系统中的知识项看成对象项  $E$ , 且对  $E$  经过一定的操作产生核  $t^c$  和包络  $t^e$ , 从而形成该问题的一个粗集, 构成不确定性区间。

考虑知识系统中对象项的集合  $E = \{e_1, e_2, \dots, e_m\}$ 。例如, 有疾病诊断系统的症候作为对象集, 即  $E = \{\text{头痛, 发烧, 非柯氏斑, 痊子}\}$ , 存在有假设集  $T = \{t_1, t_2, \dots, t_p\}$ , 比如  $T = \{\text{麻疹, 结核, 百日咳}\}$ 。显然, 这个假设集是通过观察, 参照症候对象集  $E$  而得到的。例如有诊断规则  $t_1 \rightarrow e_1 \wedge e_2$ , 它的一个可能解释是“麻疹蕴含发烧和非柯氏斑”, 即有  $t_1 \rightarrow e_2 \wedge e_3$ 。

再考虑概念集  $A = (a_1, a_2, \dots, a_n)$ , 对于一个知识系统而言, 概念  $a_i (i=1, 2, \dots, n)$  是领域知

识中的基本属性,来自于领域中的每条信息都存在有与  $a_i$  的确切关系。然而,它们并不都一对一地映射到  $e_i$  和  $t_k$ ,其中有些  $a_i$  是未知的,有些已知的  $a_i$  与  $e_i$  相关联,存在有  $e_i$  上的映射,同时也存在有  $t_k$  上的映射。例如,关于“麻疹”的症候规则也许是:

$$a_1 \wedge a_5 \rightarrow a_1 \wedge a_2 \wedge a_3 \quad (6-1)$$

这时,由于观察到的事实在这些概念  $a_i (i=1,2,3,4,5)$  上没有明显的映射关系,医生仅根据所观察到的事实陈述病症,所以这个陈述仅仅只有一种可能。这就是说  $t_k \rightarrow e_i \wedge e_j$ ,其中  $e_i = a_2, e_j = a_3 \wedge a_6, t_k = a_4$ ,那么有

$$a_4 \rightarrow a_1 \wedge a_3 \wedge a_6 \quad (6-2)$$

显然,(6-1)式与(6-2)式不同。这说明应用观察事实描述系统,一般来说仅仅有可能逼近所隐藏的概念。RST 用于不确定性表示的优点在于它可以跟踪这一逼近的过程。因为粗集中  $e_i$  和  $t_k$  对于  $a_i$  有关联,如果  $e_i$  和  $t_k$  是由  $a_i$  得到的话,那么就可以写出关于它们的逻辑表达式,相应也就可以写出基于领域专家知识未知集  $A$  的  $e_i$  和  $t_k$  之间的逻辑关系。

$A$  和  $E$  上的对象  $t$  的大小可以用它的核  $t^c$  和包络  $t^e$  的基来度量,即

- 若  $t=t^c$ ,那么  $A$  由  $(A, E)$  精确定义
- 若  $t \neq t^c$ ,且  $t^c \neq \emptyset$ ,那么  $A$  由  $(A, E)$  粗集定义
- 若  $t^c=\emptyset$ ,那么  $A$  无  $(A, E)$  的下限定义,即粗集无核
- 若  $t^c=E$ ,那么  $A$  无  $(A, E)$  的上限定义,即粗集无包络
- 若  $t^c=\emptyset$  且  $t^c=E$ ,那么  $A$  完全不能由  $(A, E)$  定义,即既无核又无包络。

## 习题六

1. 分别以不精确性、不完全性、模糊性和非单调性举例说明现实世界知识的不确定性。

2. 构造一个不确定性知识系统一般要涉及几个问题?

3. 设某问题求解用到下面的推理规则:

- Rule 1: IF  $E_1$  THEN  $H_1(0.9)$
- Rule 2: IF  $E_2$  THEN  $H_1(0.5)$
- Rule 3: IF  $E_3$  and  $E_4$  THEN  $E_1(0.8)$
- Rule 4: IF  $E_5$  and  $E_6$  THEN  $E_2(1.0)$

推理网络图如图6-7所示。试用确定因子法求出  $H_1$  的可信度  $CF(H_1, E_1 \& E_2)$ ,假定已知在当前观察有  $CF(E_3)=0.3, CF(E_4)=0.4, CF(E_5)=0.5, CF(E_6)=0.6$ 。

4. 设有一组规则

- Rule 1: IF  $E_1$  THEN  $H(0.8)$
- Rule 2: IF  $E_2$  THEN  $H(0.6)$
- Rule 3: IF  $E_3$  THEN  $H(-0.7)$
- Rule 4: IF  $E_4$  and  $E_5$  THEN  $E_1(0.7)$
- Rule 5: IF  $E_6$  and  $E_7$  THEN  $E_2(1.0)$

推理网络如图6-8所示。现已知  $CF(E_3)=0.2, CF(E_4)=0.8, CF(E_5)=0.7, CF(E_6)=0.8, E_7 = E_8 OR E_9, CF(E_8)=0.4, CF(E_9)=0.9$ ,求结论  $H$  的不确定性。

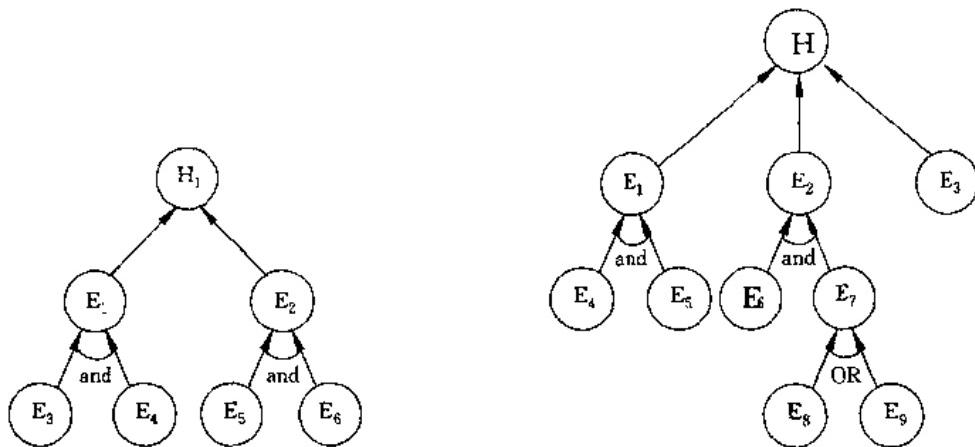


图 6-7 习题 6-3 推理网络

图 6-8 习题 6-4 推理网络

5. 设有下面规则：

- Rule 1: IF  $A$  THEN  $B(20, 1)$
- Rule 2: IF  $A_2$  THEN  $B(300, 1)$
- Rule 3: IF  $A_3$  THEN  $B(75, 1)$
- Rule 4: IF  $A_4$  THEN  $B(4, 1)$

当证据  $A_1, A_2, A_3$  和  $A_4$  必然发生后,  $B$  的概率如何变化?(已知  $B$  的先验概率为 0.03)。

6. 图 6-9 所示的推理网络取自专家系统 PROSPECTOR, 试利用该系统的概率推理方法, 计算假设 HYPE 的后验概率。其中 FMGS、PT、RCIB 为初始证据。网络结点右上方数字为先验概率, 结点连线处的一对数字为相应的 LS 和 LN。

设  $s_i$  表示观察证据, 且用户告知

$$\begin{aligned} C(\text{FMGS} | s_1) &= 3 \\ C(\text{PT} | s_2) &= 1 \\ C(\text{RCIB} | s_3) &= -2 \end{aligned}$$

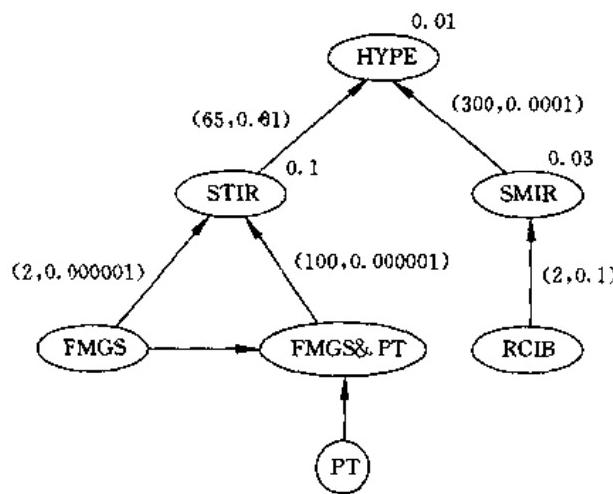


图 6-9 习题 6-6 推理网络

7. 设有下面规则

Rule 1: IF  $E_1 \wedge E_2$  THEN  $E_3 = \{a_1, a_2\}$  CF = {0.2, 0.6}

Rule 2: IF  $E_5 \wedge E_6$  THEN  $E_4 = \{b\}$  CF = {0.8}

Rule 3: IF  $E_3$  THEN  $H = \{h_1, h_2, h_3\}$  CF = {0.2, 0.5, 0.3}

Rule 4: IF  $E_4$  THEN  $H = \{h_1, h_2, h_3\}$  CF = {0.3, 0.4, 0.1}

已知:  $CER(E_1) = 0.7, CER(E_2) = 0.8, CER(E_5) = 0.6, CER(E_6) = 0.4, CER(E_7) = 0.9, E_6 = E_6 \text{ OR } E_7$ , 并假定:  $|U| = 20$ , 求  $CER(H)$ 。

(推理网络如图 6-10 所示)

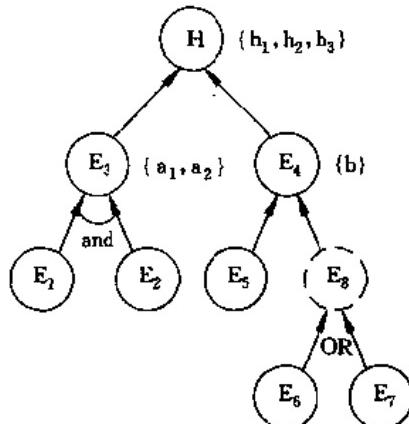


图 6-10 习题 6-7 推理网络

8. 设有辨别框  $\theta = \{A, B, C\}$ , 对于下面给出的两个  $m$  函数

|       | $\{A, B, C\}$ | $\{A, B\}$ | $\{A, C\}$ | $\{B, C\}$ | $\{A\}$ | $\{B\}$ | $\{C\}$ | $\emptyset$ |
|-------|---------------|------------|------------|------------|---------|---------|---------|-------------|
| $m_1$ | 0.15          | 0.3        | 0.1        | 0.1        | 0.05    | 0.2     | 0.1     | 0           |
| $m_2$ | 0.2           | 0.1        | 0.05       | 0.1        | 0.3     | 0.05    | 0.2     | 0           |

(1) 分别计算  $m_1$  和  $m_2$  对应的信任函数和似然函数以及各个集合的置信区间。

(2) 计算两个证据组合的 bpa 函数、信任函数、似然函数和相应的置信区间。

9. 设有如下推理规则:

Rule 1: IF  $E_1 \text{ AND } E_2$  THEN  $A = \{a_1\}$  CF = {0.8}

Rule 2: IF  $E_2 \text{ AND } (E_3 \text{ OR } E_4)$  THEN  $B = \{b_1, b_2\}$  CF = {0.4, 0.5}

Rule 3: IF  $A$  THEN  $H = \{h_1, h_2, h_3\}$  CF = {0.2, 0.3, 0.4}

Rule 4: IF  $B$  THEN  $H = \{h_1, h_2, h_3\}$  CF = {0.3, 0.2, 0.1}

初始证据的确定性为:

$$CER(E_3) = 0.7, CER(E_4) = 0.8, CER(E_5) = 0.9$$

假设  $|U| = 10$ , 试画出推理网络图并求出  $CER(H)$ 。

10. 设有模糊集合  $U = \{1, 2, 3, 4, 5\}$  和模糊规则:

IF  $x$  is 低 THEN  $y$  is 高

其中“低”和“高”均为  $U$  上的模糊子集, 并设为

$$\text{低} = 0.9/1 + 0.7/2$$

$$\text{高} = 0.3/3 + 0.7/4 + 0.9/5$$

已知事实为  $x$  is 较低

“较低”的模糊子集为

$$\text{较低} = 0.8/1 + 0.5/2 + 0.3/3$$

试用可能性理论求出模糊结论。

11. 考虑早晨找衣服穿的问题,为此必须利用下面的一些知识:

- (1) 穿工作服,除非它们是脏的,或者
- (2) 如果天气寒冷,那么穿毛衣
- (3) 通常冬天气是寒冷的
- (4) 如果天气暖和,则穿 T 恤衫
- (5) 通常夏天天气是暖和的。

试为了解题所需的事,建立一个 TMS 式的数据库。

12. 有人发现缺省逻辑和自认识逻辑的推理机制十分相近,请看下表

| 缺省逻辑           | 自认识逻辑            |
|----------------|------------------|
| 已知 $A$ 为真,     | 我相信 $A$ 为真,      |
| 没有证据说明 $B$ 为假, | 没有根据使我相信 $B$ 为假, |
| 则 $C$ 为真       | 则我期待 $C$ 为真      |

请研究并证明,在什么意义下可以认为这两种逻辑等价。

13. 能否用 McCarthy 的限定理论表达下面的命题。

- (1) 李敏之只要有一次考试得优,以后的任何考试必得优。
- (2) 只要有一人考试得优,任何人考试都能得优。
- (3) 任何人只要一次考试得优,那么以后考试都会得优。

## 第七章 规划求解系统

与需求解的一般典型问题相比,人工智能规划求解系统往往注重大型复杂对象的研究。前者关心的是具体问题的具体解答结果,而后者注重的是系统求解过程及解路径(solution path)的寻取。

其次,人工智能研究的传统规划问题,关注高层次求解系统技术的自动实现,重视“拟人”智能特性的发挥,使系统能对现场具体任务和环境,自动进行分析与描述,自动选择求解步骤,形象生动地实现其过程。例如机器人规划(robot planning),计算机下棋(computer chess)等。

应该指出,目前关于规划求解问题,客观上已形成了两分支发展的格局:一条是人工智能科学关于研究自动规划系统及其实现技术的道路,另一条则是管理运筹科学中关于应用数学规划的研究分支。

后者针对诸如如何合理配置使用资源,取得最优生产及经济效益等类具体问题,运用数学方法,设法建立或找到容纳若干环境参数相约束的数学模型,并依照实验不断修正模型,最后依靠模型计算,寻取本原问题最优解。

如何将上述两种不同的研究思路互取所长,协同发展呢?二者如何兼容并蓄,纵横综合,以便共展规划科学的新蓝图呢?这或许正是新时代研究规划科学工作者的又一新课题。

本章将介绍并深入探讨人工智能规划求解问题的原理、方法和技术。

### 第一节 规 划

#### 一、规划的概念

规划(planning),可认为是循规策划的缩略词。意即在行动之前,确定达到系统目标的进程。换言之,是寻求快速到达目标的步骤和路径。

究其规划内涵,具有两层含义:既要遵循客观规律,确立到达系统目标的方略计划,又须依照一定的技术方法,实施运作步骤,力求取得最佳解操作序列。简言之,规划就是制定、实施数行动的步骤与决策。

一个规划,就是一个行动过程的描述。如图7-1所示结构,是一位大学生一天正常的作息安排,它就是一个规划构或。

图7-1的例子还简要表现出:一个总规划,可以含有若干子规划和求解流程的结构。

#### 二、规划的特性和作用

缺乏规划,将导致求解效率极大程度的降低。例如,有人为了买邮票和寄信,因为缺少规划,跑了两次邮局。

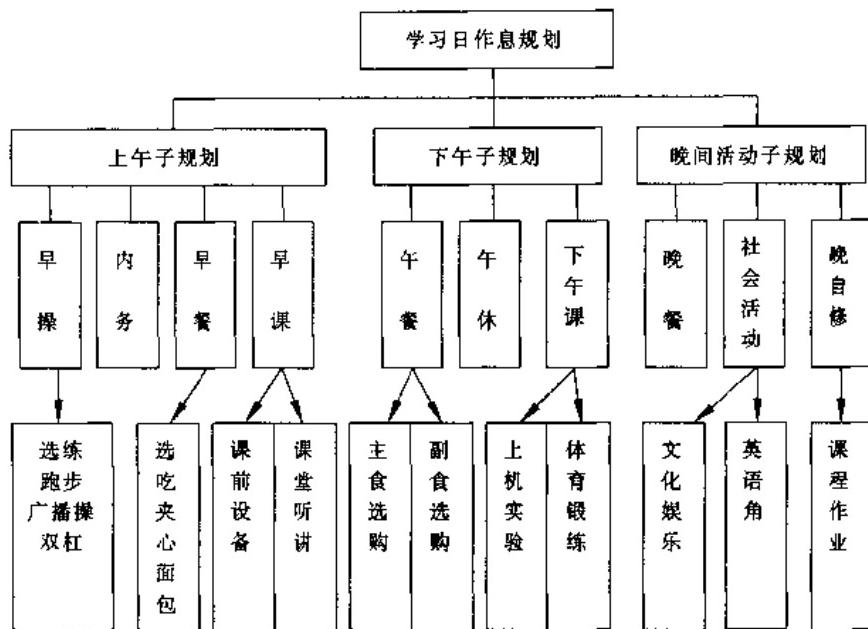


图 7-1 大学生学习日作息规划

在问题规划求解中,规划表现出既有相互独立,多条局部解路径可选的特性,又包含序列相贯,不可随意颠倒顺序的环节特性。例如,走出大森林,到达目的地,可以有多条路线,但每条路线所须跨越的山谷川流的次序大致都是固定的。

规划的环节特性,常表现为若干相关子规划首尾相串的进程顺序,执行前一子规划所获取的子目标,是正确实施后一子规划的前提条件,简称为约束,如图7-2所示。图中弧线表示总规划分解为相关子规划的合取关系。

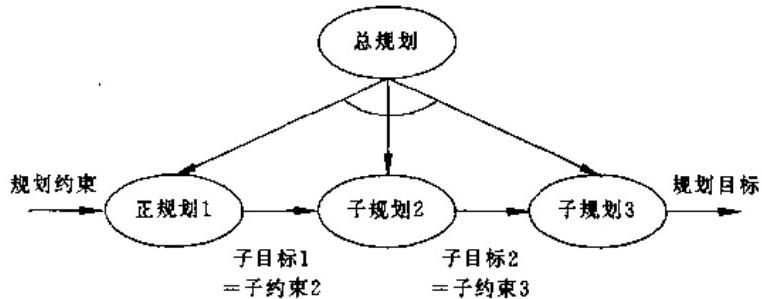


图 7-2 规划的相关性

例如,欲建一座优质大楼,规划环节应包括:预先设计图纸,施工先须打好地基,预留地下管道等。地下工程完毕,再实施地上工程,先起地面楼层,再逐步续起高层。但是,如果缺乏合理规划,工程施工颠倒了次序,该工程就将失败或造成极大损失。

按规划实施,还可以监控系统运行进程。这样,可以及时发现并避免差错。例如,考虑某个在深海打捞作业的潜水机器人,它必须能够规划一个探测海域环境,随时掌握现场作业面的步骤和进程。当发生深海潜流影响,导致目标物位置变动时,机器人能根据检测到的环境参数与预期不符的差异,在向海面指挥中心发出告警信息的同时,还须启动作业修改进程规划,采取必要的调整作业进程步骤以及操作措施。

### 三、系统规划求解的方法与途径

把大型复杂的系统的规划求解,化为若干复杂度低得多的子系统、子目标寻优过程的综合作用,这种降低复杂问题求解难度的分析方法,称为分解技术。它是实现规划求解的主要技术手段。按照采用规划决策的不同思维方式,规划技术可以有许多分解途径。图7-3列出了部分规划分解的技术实施方案。

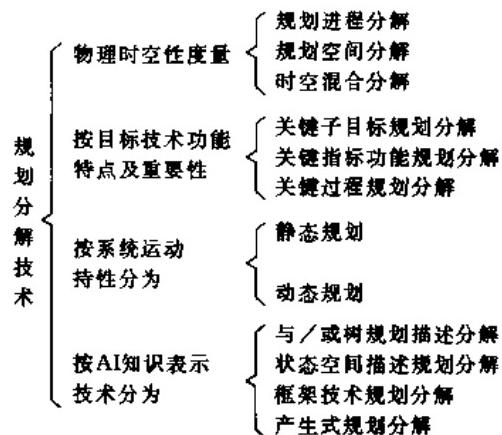


图7-3 问题规划分析的技术途径

如图7-4所示的规划与/或树,是人们常喜欢采用的一种简化规划求解难度的重要思想。图中,问题的难度逐层降低。

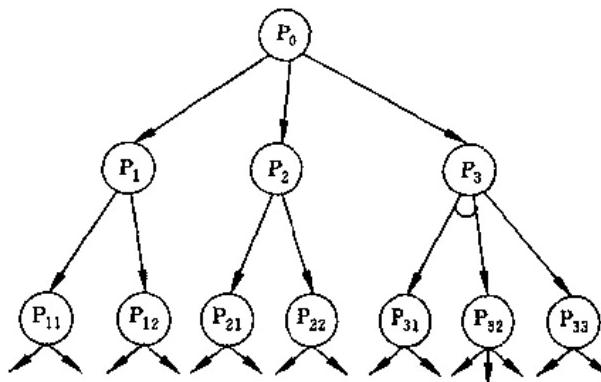


图7-4 问题求解规划与/或树

注： $P_0 = P_1 \wedge P_2 \wedge P_3$      $P_1 = P_{11} \vee P_{12}$      $P_2 = P_{21} \wedge P_{22}$      $P_3 = (P_{31} \wedge P_{22}) \vee P_{33}$

## 第二节 机器规划成功性基本原理

### 一、概述

实现机器智能的自动规划系统,存在许多非本原问题的因素。对机器规划成功性的考察,更重要的是从历史性发展,总体性和心理学上去探询认识。机器自动规划问题,是现实科技领

域中高难度问题。现实中人类智能规划,尚难保证不出差错,机器智能规划又岂能轻易成功?!事实上,规划的研究不在于一次性获取成功,而在于可从若干次不成功中,进行机器归纳学习,使得修定的规划比以前任何一次都更加接近成功。

近年来,随着计算机硬件与软件技术的飞速发展,特别是人工智能技术在各行各业应用的不断深入,在机器自动规划方面已取得了许多重大成果。例如,在海湾战争中,美国军方利用计算机规划运送作战物资方案;1997年5月美国IBM公司“深蓝”(Deep Blue)计算机,战胜国际象棋世界冠军加里·卡斯帕罗夫(Garry Kasparov),首次登上了国际象棋世界冠军的宝座;1976年,美国数学家Appel和Haken,运用计算机辅助证明(CAP)的规划思想,一举攻克了世界三大数学难题之一的“四色定理”的证明。至于利用多媒体计算机音响装置,让计算机弹奏出具有随机数字符号音乐,人们已是司空见惯。

上述事例说明,机器自动规划全面成功地解决,虽然很难,但又是必然的。人类应该保持平常心态,欢迎这一划时代的来临:正如汽车比人跑得快,起重机比人举得重,计算机比人算得快,机器人可能将在若干方面超过人的能力,这又有什么稀奇呢?

综上所述,面对机器自动规划系统成功性的研究,尚需研讨下述若干具体问题:

- (1)如何进行问题总规划的设计?
- (2)当规划失败时,如何确诊规划失败的环节,并进行规划修正?
- (3)如何进行机器规划的自学习?如何自动生成规划系统及过程?

诸如上述问题的研究,实际上当前只能结合具体规划生成系统,针对具体问题环境,进行具体分析和设计。下面仅就前两个问题,综合进行原则性的讨论分析,在此基础上,介绍若干基本原理。

## 二、总规划的设计与分层规划原理

普遍常用的原则是:先产生一个粗略的规划,再设法将它细化成若干较简单的基本级规划;依此类推,逐级分层,下级比上一级更精细,形成多级分层规划(如图7-5所示)。

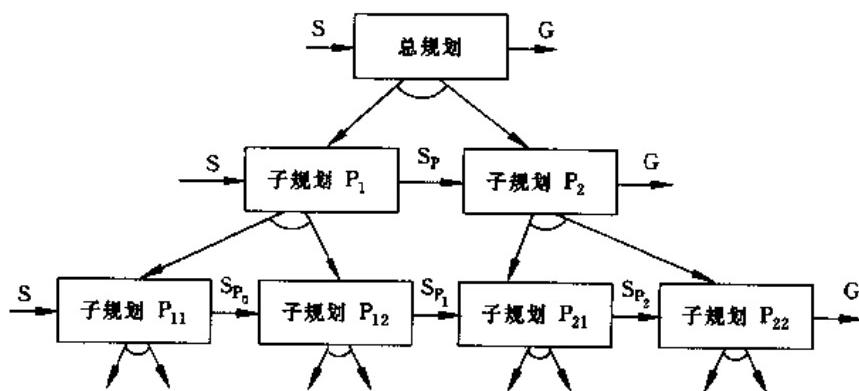


图7-5 多级分层规划

注:S称为规划初态集,G称为规划目标集  $S_P, S_{P_1}, S_{P_2}, \dots$  称为子目标状态集

若单纯从多级分层规划结构图来考察,不难发现该结构图客观上具有这样的一种特性:若某规划的各级子规划均有解,则表明该规划已可解;相反,若高级规划有解,则并不表明其随机划分的低级子规划也一定有解。事实上,按照分层规划思想,既然高级规划已可解,人们也无

须对它再进行低级子规划的分解了。

综上所述,规划成功可解的原理可简叙为:在多级分层规划中,只要任意找到一条从初态集到目标集的可解路径,则该规划可解;否则该规划无解。上述原理的正确性验证显而易见,此处不再赘述。

### 三、规划问题求解与最优规划原理

规划问题求解,一般应包括寻取解路径和解结果两个方面。而规划最优解问题则可概括为寻取其中之一。因为最佳解路径和最佳解结果是相互唯一对应的,故知其一就可立即得到另一个。

若最佳解路径是所有解路中深度最小的,且对应于操作序列所使用的策略集是最精练的,即数量上是最少的。

设  $D_0$  为总规划策略集,  $S_P$  为规划最佳解路径上任一中间结点,  $D_P$  为已使用的子策略集,  $S$  为规划的初态集,  $G$  为目标集。则最优规划原理可叙述为:若  $D_0$  为最优规划策略集, 则余下的  $S_P \xrightarrow{D_G} G$  策略子集  $D_G = (D_0 - D_P)$  也必是最优的。

换言之,一个最优规划策略集的余子集总是最优的。

上述最优性规划原理也可称作最优性规划定理,且可作为操作规则运用子规划问题求解中。

上述定理,可用常识性推理论证如下:

已知  $D_0$  是最优的,依照最优规划策略集取得最小值的前提,若  $S \xrightarrow{D_P} S_P$  已使用的不一定是最优的,则  $S \rightarrow S_P$  必定存在一个对应最佳路径的最优策略子集,且有

$$D'_P \leq D_P$$

若设  $D_P$  不一定是最优的,则  $G$  也必定存在一个最优策略子集  $D'_P$ ,于是则有

$$D'_P \leq D_P$$

若设  $D_S$  不一定是最优的,则  $S_P \rightarrow G$  也必定存在一个最优策略子集  $D'_S$ ,于是有

$$D'_S = D_0 - D'_P \geq D_0 - D_P$$

而由已知

$$D_S = D_0 - D_P \leq D_0 - D'_P = D'_S$$

得到

$$D_S \leq D'_S$$

的结论,这显然与  $D_S$  不为最优子集假设矛盾,故只可能  $D_S = D'_S$  为  $S_P \xrightarrow{D_S} G$  对应于最佳解路径的最优策略子集。

同理,还进一步说明了  $D_P$  也必定是  $S \rightarrow S_P$  的最优策略子集。需要强调的是,这里提出关于最优规划原理及其证明的前提虽然不十分严密,但仍不失为既具理论意义,又有极强的实践指导意义的分析研究方法。这是因为:建立在极值条件下,可找到最优解相关判据,它是科学规律中一般自然法则。

当然,针对具体自动规划系统,应用最优规划原理分析,将成为要考察多参数、多变量、多极值方向条件下,确定最佳解路径的复杂问题。尽管如此,依照相关泛函知识和最优规划原理综合分析,仍可导出相关规划和推理。

## 四、规划的双序求解与诊断

根据问题规划求解的方向,若从初始起点出发,逐级向前推理,直至获得到达目标的序列操作,并能实现终点的全部状态条件。系统这种按顺序方式求得解路的过程,称为顺序规划求解。反之,从目标终点逆向推理,直至系统达到与初态一致的约束条件。这种反向求解过程,称为逆序规划求解。顺序求解和逆序求解统称为双序法。

顺序规划求解是一种顺其自然方式的常规方法。而逆序规划,依据目标要求来验证前提条件;或根据目的分析,确定必要的相应手段与操作规则,这种思路似乎更加符合人类思维习惯,且又称之为“目的手段”规划分析法。

例如,警卫机器人的行动规划之一:机器人为了照亮房间,启动了电灯开关,为了对付入侵者,又拿起了武器;并对准猎物,发出了要开枪的警告……。

人们还发现很多情况下,逆序法比顺序法更易找到最佳解。例如,凡属可表示为树状性质的问题,采用逆序规划,从树叶寻其根十分容易,且往往得到的解就是最佳解。而采用顺序法,从根求其某片特别的叶子,却要困难复杂得多了。

为了诊断系统规划是否出错,通常采用逐级规划分段测试法和双序法相结合的方式进行。针对某级子规划,可先选取逆序检查法;若逆序不成功,再用顺序法进行。只要其中一个方向成功,则称该级子规划可解。否则,需进一步实施微细诊断或出错修正。

不同的规划系统,优先选用顺序法还是逆序法,要视具体情况分析再决策。例如,规划工程实验及设备检修,人们常运用输入信号波形,逐点检测追踪的原理,实质上即选用分段诊断和顺序查障法,这样更便于快速查找出故障段点;而专利技术设计规划,往往多用逆序思维方式进行。

## 第三节 规划搜索求解

### 一、搜索域的分层规划

为从大型复杂问题状态空间中,加速搜索,找到从初始状态  $S_0$  到达目标状态  $S_g$  的解,这时,可利用问题的相关知识和启发信息,在  $S_0$  到  $S_g$  之间依序找出并选取若干关键状态  $S_k (k=1, 2, \dots, N)$ , 把  $S_k$  作为从  $S_0 \rightarrow S_g$  的解路径中的子目标, 形成  $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow \dots \rightarrow S_N \rightarrow S_g$  层层搜索, 逐次逼近总目标的求解格局。这种依照多级分层规划思想, 通过中间序列关键状态, 采取层层子目标任务递进的手段, 达到快速顺利完成总目标搜索任务的技术, 称为分层规划搜索技术。该方法又称为规划搜索的关键状态归约法。

设原始大型问题的状态空间表示为三重序元组:

$$T = (S_0, F, S_g)$$

式中: $S_0$  为初始状态, $S_g$  为目标状态, $F$  为  $S_0$  到达  $S_g$  的可供应用的操作集。分解为子规划问题的各状态空间为:

$$\begin{aligned} & (S_0, F_1, S_1), (S_1, F_2, S_2), \dots, (S_{k-1}, F_k, S_k) \\ & \dots, (S_{N-1}, F_N, S_N), (S_N, F_g, S_g) \end{aligned}$$

式中,  $S_k$  ( $k=I, II, \dots, N$ ) 为所引入的若干中间关键状态, 并选作为  $N$  个子规划的子目标;  
 $F_k$  ( $k=I, II, \dots, N$ ) 为对应于子规划的操作集子集, 且  $F_k \in F$ .

应用分层规划的关键状态归纳法, 具有下述特点:

首先, 可以大幅压缩搜索空间。如图 7-6 所示,  $B$  —— 原始大问题  $(S_0, F, S_g)$  的全搜索空间;

$B_k$  —— 子规划问题  $(S_{k-1}, F_k, S_k)$  的搜索空间。由图可见,  $\sum_{k=1}^{N+1} B_k \ll B$ ,

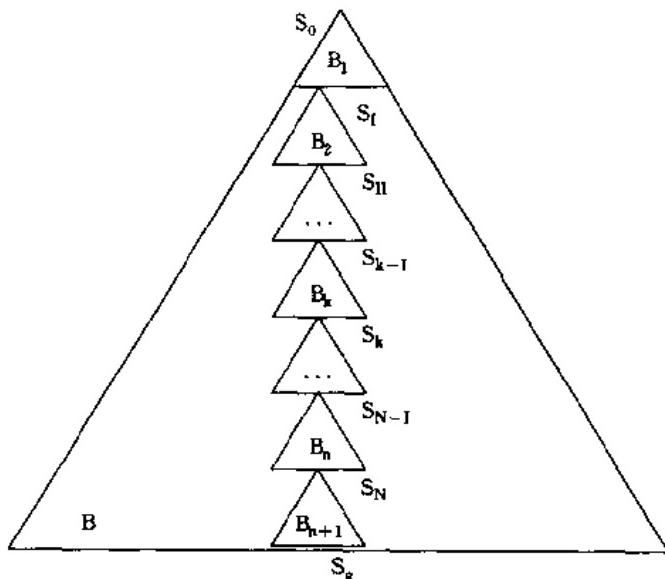


图 7-6 分层规划搜索

其次, 对子规划问题的求解, 便于在小范围内应用相关经验和知识等启发信息, 提高搜索效率。

另外, 若有些子问题已有成熟解法或已为本原问题, 则更有利子加快搜索求解过程。例如, 已知用 1 艘船可分别解决 2 人与 2 野人或 3 人与 3 野人的规划过河问题。现问用 2 艘船能否解决 4 人与 4 野人或 6 人与 6 野人的过河问题。采用分解规划, 两船同步行动, 则该问题即可利用已有解的本原子问题快速获解。

## 二、分层规划搜索的并行处理

并行处理, 是多个子系统同时启动的处理工作方式。它以并行方式, 增大操作空间。换取时间的缩短, 是一种加快全系统搜索的有效手段。

例如, 为了迅速从海域中搜索一艘行船, 可派多个机器船及飞行器, 分别在不同的海区同时进行扫描和搜索。

直接并行处理, 适于相关子规划初始条件相对独立, 且可满足的搜索中。

对于那些前后次序不可颠倒, 且各子问题特性不一的批量子系统的规划搜索, 不能直接应用上述处理方式。这时可采用一种称为流水线的工作方式, 使多个资源及功能操作部件各司其职, 并行协调, 以便最好发挥系统的速度和效率。

例如, 在机器人工厂中, 一件产品, 需  $2n$  道工序子规划, 每工序由功能专一的机械手或机器人担当。操作中有一半工序各需耗工时  $t_0$ , 另一半各需工时  $2t_0$ 。现需批量制造该产品, 问如

何配置机器人资源及安排岗位,可使规划搜索系统获最高效率?

这时可按照生产工序顺序排列,成为流水线作业方式,配置机器人资源及安排岗位可采用并行处理和流水线相结合的作业规划进行;凡耗时为 $t_0$ 的工序各指派一台机器人担当;凡耗时为 $2t_0$ 的工序,则选派2台机器人共同轮作。这样,很容易算得该流水线周期为 $t_0$ ,即从第一件产品输出后,只需每经过 $t_0$ 的时间间隔,系统就产出一件新产品。

### 三、一个实例——魔方问题的规划搜索求解

魔方是一种智力搜索解谜玩具,曾经风行全球,流传很快很广。据说,这是由匈牙利建筑学家厄尔诺·鲁比克设计出来的。如图7-7所示,这是一种由26块小立方体组合而成的彩色六面体,每个面都可围绕相应垂直中心轴任意转动,从而变动图案。

在26块小立方体中,每个面的中心块,即上(U)、下(D)、前(F)、后(B)、左(L)、右(R)6个中心块的位置不变;其余8个角块,12个边块都可以分别在绕轴转动中,随意改变位置和方向。

根据排列组合计算,这20个可动立方体共有约 $4.3 \times 10^{19}$ 种选择排列,因此要把魔方中任意一种状态变成某种目的状态时,就需要在 $4.3 \times 10^{19}$ 种可能中去搜索,可见直接求解该问题,颇有难度。用分层规划思想,求解该问题是一种有效手段。例如,美国人杰·诺尔斯设计了一种五阶段魔方求解法,步骤如下:

- (a)按目标状态调整好上边块的位置和颜色取向标志;
- (b)按目标状态调整好上边块及上角块的位置和颜色取向;
- (c)按目标状态注意设法保持上述成果同时,调整好中边块的位置颜色取向;
- (d)按目标状态调整底角块位置、颜色的取向,注意调整变换中设法恢复上述成果的已有状态;
- (e)按目标状态调整变换底边块,使完全达到目标状态。

在上述变换规划搜索步骤中,愈接近目标状态时,随着须注意保持已成功的成果,记忆调整难度也随之加大。但由于上述操作每一步骤主要只涉及4个小立方体,使其可能的排列组合状态数大为减少,加之已有成熟算法求解每个子规划搜索,故魔方问题通过规划搜索可以获取成功解决。

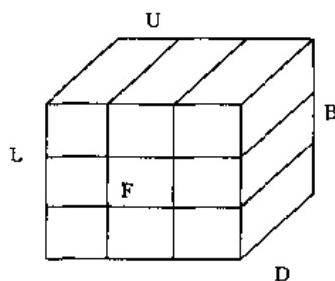


图7-7 魔方:一种智力玩具

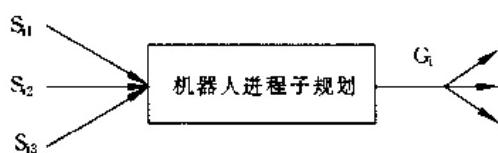
### 第四节 机器人规划问题求解

指派机器人自动选择若干操作,制定并执行一个动作序列规划,代替人去完成给定的任务。例如,要求机器人完成焊接,进行特殊装配等,这类机器人,在实际中已有应用。尤其是人难以进入的恶劣环境,机器人却不惧怕。例如,充满有害气体的地下坑道,或是有放射性污染试验场地,指派机器人代替人去搬运一些小物体等,这类应用更具有特殊意义。

## 一、机器人工作规划及生成方法

应用分层规划技术,可将机器人须完成的总任务,分解为若干依序排列的子任务。总任务即总规划,子任务即子规划,又称为一个过程。

机器人在满足相应约束条件下,选择操作序列,按照子任务目标的逐一实现,逐步达到总任务目标。这个总任务完成及其指令操作序列的生成全部进程,称为机器人工作规划。其中对每一个进程,还须仔细生成子任务工作规划,如图7-8所示。



图中, $S_{ij}$ 称为输入约束条件, $\sum_j S_{ij}$ 即为进程的全部初始条件; $G_t$ 表示为该进程的目标状态及其输出条件。

图7-8 机器人进程子规划

生成进程子规划的主要方法之一,即采用关键状态归约法。该方法要求依据任务进程分析,

选取并确定若干关键中间状态,并把这些状态对应选为进程子规划的目标,依序分别规划求解,直至总目标任务全部完成。

例如,机器人搬运物件,一般必须有3个进程:

进程1), 机器人须到达要搬运的物件对象存放地;

进程2), 机器人抓取并携带物件走至安放物件所在位置;

进程3), 机器人放置物件后,按指令回到指定地点。

上述3个进程若有颠倒或相关条件不满足,就会导致执行规划的失败。

制定或执行进程子规划时,很容易发现:前一进程子规划的目标,恰恰就是启动紧邻后进程规划的初始条件,进程的这一属性,又称为进程的链接特性。

生成子规划的另一重要途径,则是依据关键操作确定进程目标和步骤的方法。

例如上例中,按关键操作发生的先后顺序,可把机器人搬运物件分为4个进程:

进程1), 机器人走到物件源存放处;

进程2), 机器人在物件源存放处抓取到物件;

进程3), 机器人走向并走到物件安放处;

进程4), 机器人在指定处准确放下抓取物。

上述按关键操作,确定发生的状态变迁,分析并生成进程子规划的方法,称为关键操作归约法。

## 二、规划的执行与操作控制

当系统规划一旦生成,机器人就可以按照指令,顺序执行该规划步骤和进程。实际上,机器人执行规划,就是机器人不断选取操作,执行操作的过程。当全部操作序列执行完毕,达到目标状态。这时,如果没有新的指令,机器人就应能自动停下来。

但是,在机器人执行规划中,如何能保证系统自身能准确执行全部规定的操作,并能达到预期效果呢?

事实上,仅具有开环控制功能的规划系统,一般难以察觉出现的微小偏差。为避免这种偏

差造成系统的失误,可在机器人规划中加进闭环控制,这是保证机器人可靠执行规划的关键方法和技术(如图7-9所示)。

例如,搬运机器人到达物件源存放处时,工作台上出现了被调换的另一类危险物品,这时机器人是否仍然发出抓取物件动作呢?显然,如果机器人规划缺少闭环监测和识别控制功能时,就可能发生抓错物件的事故。而对于具有视觉及检测识别规划的机器人来说,系统就可以避免这类风险。

### 三、机器人过程控制与环境约束

通常,机器人选择执行一项操作时,首先总要检测当前状态,是否满足该项操作所要求的约束条件,这一比较过程称为匹配(matching)。若状态条件满足,即匹配成功,则执行相应选定的操作。此时,须检测并验证操作前后状态变化是否正确合理,从而删除操作前相关状态,跃迁到新的状态。另一方面,当发现不能匹配时,就应放弃执行原选定的操作或转入相关处理。

事实上,如果比较机器人在执行某项操作前后的环境约束及状态条件,则不难发现:机器人每次执行操作,实际上只是改变了极个别相关对象的状态,而大部分的环境状态和约束条件则不发生变化。因此,在机器人规划及进行操作条件过程中,可利用状态变迁中的这一特性,仅对动态变化的部分状态空间予以优先重点考察,面对大部分静态约束条件只需予以标识保留。这样处理的好处在于,对机器人相关规划操作,能快速进行条件选择与匹配,提高规划求解效率。例如,考察机器人装配工作间环境,墙上贴有若干图片,还安放有电子挂钟和电铃;窗台上上有若干盆花,房间一角有吸尘器、电话、水龙头……等。但这些环境因素并不对机器人安装规划产生直接影响。因此,在过程规划描述中,可将其略去,以便减少不必要的信息检测与处理。

## 第五节 基于谓词逻辑的机器规划设计

谓词逻辑描述作为一种人—机交互的语言形态,因其具有良好的界面特性,在人工智能研究中应用很广。

### 一、机器人行动规划问题分析

如图7-10所示,设在一含有四室房间内,机器人小姐玛丽(Mary)正在紧张工作。*a* 处和*b* 处各有一茶桌(Teatable),其中 *a* 桌放有一只花瓶,花瓶中插有一束鲜花;*b* 处有刚刚进门入座的贵宾罗西(Roxi)。现在要求机器人小姐完成的主要任务是:为表达本公司对客人的欢迎,立即把花献给人座的贵宾前。要求规划机器人这一行动过程。

这一过程虽然可以采用搜索、语义网络、框架等方法描述求解。比较起来,选用谓词逻辑规划求解,可以生动拟人动作过程。首先分析行动过程并规划关键步骤,主要有:

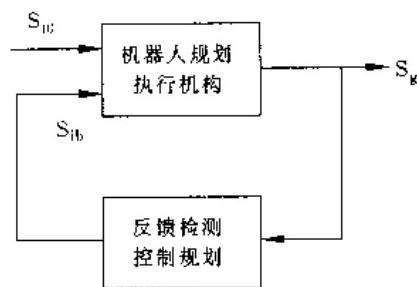


图7-9 具有反馈控制规划的机器人执行机构

图中: $S_{i0}$ —输入约束条件, $S_k$ —输出的状态条件, $S_{ib}$ —检测输出的监控信息条件

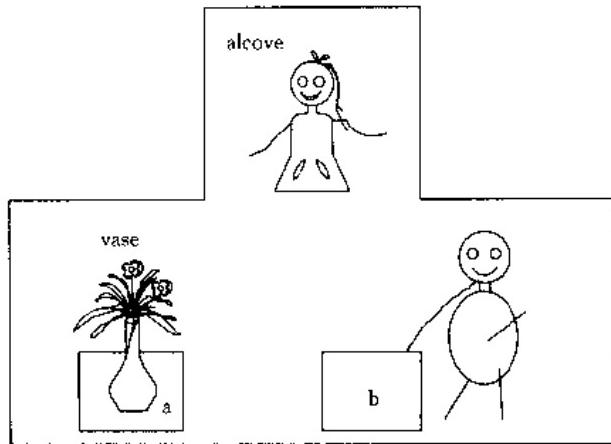


图7-10 机器人行动规划问题

步骤1，描述房间环境状态，用以表达机器人工作现场背景及变迁过程状态迁移。其中尤其包括描述初始状态及确定目标状态。

步骤2，确定机器人需进行的若干关键操作。机器人可供选择的操作有语言、表情、动作等，其中行动动作是关键。而可选择的动作有行走、抓物、放物等。关键操作的选择需要依据行动的目的与效果的分析。

步骤3，确定实施关键操作的地点、对象及进行先后顺序，匹配规划等。这是一个更为复杂的综合性问题。稍有差错，还会出现荒谬的现场局面：例如，机器人动作顺序颠倒，到a处取花，却到b处打翻并取走了客人的茶杯。因此，规划操作要求准确、严密、执行操作前后，都应检查约束条件匹配状况和预期的效果是否达到。

步骤4，规划的优化和补充。例如谓词状态描述，只要求定义与要完成的任务相关部分的谓词，过多的状态和操作定义可能造成干扰和冗余，不利于系统高效运行。另一方面，由于经验不足，开始定义的谓词状态及选择谓词操作，可能不够，则可以在规划设计中进行补充和调整。

## 二、机器人行动规划状态与操作设计

在上述分析基础上，即可对图7-10机器人行动规划问题进行设计。

首先定义表达状态的谓词逻辑。

$\text{TEATABLE}(x)$ :  $x$  是茶台

$\text{CLEAR}(x)$ :  $x$  上是空的

$\text{EMPTY}(y)$ :  $y$  手中是空的

$\text{HOLDS}(y,z)$ :  $y$  手中拿着  $z$

$\text{ON}(z,x)$ :  $z$  放在  $x$  之上

$\text{AT}(y,x)$ :  $y$  在  $x$  附近

其中，各谓词变元的值域为： $x \in \{a, b, \text{alcove}, \text{vase}\}$

$y \in \{\text{mary}, \text{roxi}\}$

$z \in \{\text{vase}, \text{flower}\}$

问题的初始状态可表示为：

$S_0: AT(mary, alcove) \wedge AT(roxi, b) \wedge EMPTY(mary)$   
 $EMPTY(roxi) \wedge ON(vase, a) \wedge ON(FLOWER, VASE)$   
 $TEATABLE(a) \wedge TEATABLE(b) \wedge CLEAR(b)$

问题的目标状态为：

$S_g: AT(mary, alcove) \wedge AT(roxi, b) \wedge EMPTY(mary)$   
 $EMPTY(roxi) \wedge ON(vase, a) \wedge ON(flower, b)$   
 $TEATABLE(a) \wedge TEATABLE(b) \wedge CLEAR(vase)$

机器人行动规划的目标是通过执行一个操作序列，把问题的初始状态转换为目标状态。操作可分为约束条件和行为动作两个部分，只有满足相应的约束条件，才可选择一个对应的动作，同时，把当前状态变换为新的状态。

为此，要完成本例机器人行动规划，必须执行下述基本操作：

$f_1: GOTO(y, x)$  ——  $y$  走到  $x$  近前  
 $f_2: PICK-UP(y, z)$  ——  $y$  拿起  $z$   
 $f_3: SET-DOWN(y, z)$  ——  $y$  放下  $z$

严格来说，系统还可以规划许多附加复杂操作，例如，从礼仪角度，机器人小姐献花时应向候行礼，富有表情；贵宾接受献花出于礼貌和尊重，应该站立起来，并表示感谢等。但由于这些操作并不影响主要步骤的进行，为突出关键事件状态变迁，本例简化了若干操作行为。

三项基本谓词操作改变的状态差异如下：

$f_1: GOTO(y, x)$   
 匹配条件： $AT(y, \sim x)$   
 动作发生：删除： $AT(y, \sim x)$   
 增加： $AT(y, x)$   
 $f_2: PICK-UP(y, z)$   
 匹配条件1)： $ON(z, x) \wedge AT(y, x) \wedge EMPTY(y)$   
 动作发生，删除： $ON(z, x) \wedge EMPTY(y)$   
 增加： $CLEAR(x) \wedge HOLDS(y, z)$   
 匹配条件2)： $ON(z, w) \wedge ON(w, x) \wedge AT(y, x) \wedge EMPTY(y)$   
 动作发生，删除： $ON(z, w) \wedge EMPTY(y)$   
 增加： $CLEAR(w) \wedge HOLDS(y)$   
 $f_3: SET-DOWN(y, z)$   
 匹配条件1)： $AT(y, x) \wedge CLEAR(x) \wedge HOLDS(y, z)$   
 动作发生，删除： $CLEAR(x) \wedge HOLDS(y, z)$   
 增加： $ON(y, z) \wedge EMPTY(y)$   
 或匹配条件2)： $AT(y_1, x) \wedge AT(y_2, x) \wedge HOLDS(y_1, z) \wedge EMPTY(y_2)$   
 动作发生，删除： $HOLDS(y_1, z) \wedge EMPTY(y_2)$   
 增加： $EMPTY(y_1) \wedge HOLDS(y_2, z)$

有时为了增加规划操作可靠性，可以增加相关匹配约束条件。只有相关约束条件全部满足时，动作才会发生。

### 三、机器人行动规划过程设计

为了达到机器人行动规划的目标状态，必须逐步实现从初始状态到达目标状态的过程转换。按上例要求，该问题须依序实现下述5个子目标规划过程。

$P_1$ ：机器人执行操作  $f_1$ ，从四空走到  $a$  桌旁，以能满足机器人取花动作发生前所必须的约束条件  $S_1$ ；

$P_2$ ：机器人为了拿花，检查子目标  $S_1$  是否符合。为达到  $S_1$ ，则操作  $f_2$ ，即在  $a$  处拿到花束，达到子目标  $S_2$ ；

$P_3$ ： $S_2$  满足，则机器人执行操作  $f_3$ ，从  $a$  处走到  $b$  处，取得献花前的准备条件  $S_3$ ；

$P_4$ ：检查  $S_3$  是否满足，若达到，可执行操作  $f_4$ ，完成献花任务，达到状态  $S_4$ ；

$P_5$ ： $S_4$  满足，机器人执行操作  $f_5$ ，回归原位，达到目标状态  $S_g$ 。

上述过程可简单示意如图 7-11：

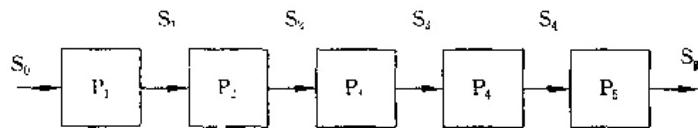


图 7-11 机器人行动规划过程

按照上述过程规划，用谓词逻辑描述的机器人行动规划过程如下：

AT(mary,alcove)  $\wedge$  AT(roxi,b)  $\wedge$  EMPTY(mary)  
EMPTY(roxi)  $\wedge$  ON(vase,a)  $\wedge$  ON(flower,vase)  
TEATABL(a)  $\wedge$  TEATABL(b)  $\wedge$  CLEAR(b)  
 $\diamond$ GOTO(mary,a)  
AT(mary,a)  $\wedge$  AT(roxi,b)  $\wedge$  EMPTY(mary)  
EMPTY(roxi)  $\wedge$  ON(vase,a)  $\wedge$  ON(flower,vase)  
TEATABL(a)  $\wedge$  TEATABL(b)  $\wedge$  CLEAR(b)  
 $\diamond$ PICK-UP(mary,flower)  
AT(mary,a)  $\wedge$  AT(roxi,b)  $\wedge$  HOLDS(mary,flower)  
EMPTY(roxi)  $\wedge$  ON(vase,a)  $\wedge$  CLEAR(vase)  
TEATABL(a)  $\wedge$  TEATABL(b)  $\wedge$  CLEAR(b)  
 $\diamond$ GOTO(mary,b)  
AT(mary,b)  $\wedge$  AT(roxi,b)  $\wedge$  HOLDS(mary,flower)  
EMPTY(roxi)  $\wedge$  ON(vase,a)  $\wedge$  CLEAR(vase)  
TEATABL(a)  $\wedge$  TEATABL(b)  $\wedge$  CLEAR(b)  
 $\diamond$ SET-DOWN(mary,flower)  
AT(mary,b)  $\wedge$  AT(roxi,b)  $\wedge$  EMPTY(mary)  
HOLDS(roxi,flower)  $\wedge$  ON(vase,a)  $\wedge$  CLEAR(vase)  
TEATABL(a)  $\wedge$  TEATABL(b)  $\wedge$  CLEAR(b)  
 $\diamond$ SET-DOWN(roxi,flower)       $\diamond$ GOTO(mary,alcove)  
AT(mary,alcove)  $\wedge$  AT(roxi,b)  $\wedge$  EMPTY(mary)

$\text{EMPTY}(\text{roxi}) \wedge \text{ON}(\text{vase}, \text{a}) \wedge \text{CLEAR}(\text{vase})$   
 $\text{TEATABLE}(\text{a}) \wedge \text{TEATABLE}(\text{b}) \wedge \text{ON}(\text{flower}, \text{b})$

程序说明：

(1)为了迅速到达目标状态,在匹配条件同时满足情况下,上述  $P_5$  过程,应用了并行处理技术:使机器人玛丽走向凹室的操作及机器人罗西(Roxi)放下花的动作并行完成,加速系统运行的效率。当然,也可以将两项操作分两步串行处理完成,直至达到目标状态结束。

(2)在程序运行过程中,有些静止状态自始至终并未改变,例如  $\text{TEATABLE}(\text{a})$  和  $\text{TEATABLE}(\text{b})$ 。如有必要,为了简化,使过程步骤更加突出明显,也可以适当加以简略。

#### 四、机器人多级规划

如图7-12所示为例,要求机器人(或机械手)完成搬积木的任务。

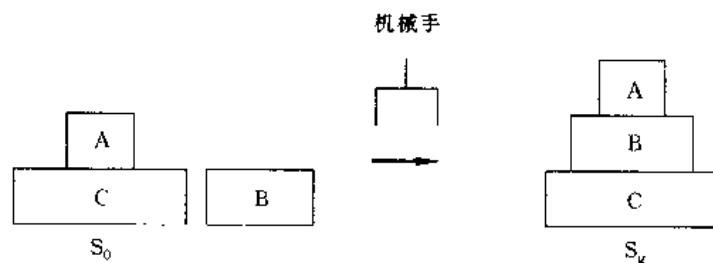


图7-12 搬积木规划问题

图中, $S_0$ 为初始状态, $S_g$ 为要求变换的目标状态,且操作中每次只能搬一块,大块不能压在小块上。该问题可用多级规划方法求解。

可列出原始问题的三级规划问题如图7-13和图7-14所示。图中,符号“ $A/B$ ”表示把积木块  $A$  放置于积木块  $B$  之上,图7-13表示清除  $B$  上的积木,其余类推。其中,图7-13表示的是对操作尚未加以修正的原始三级规划问题。在这类操作中存在着一些矛盾和冗余。例如,操作“ $B/C$ ”与操作“清  $C$ ”是相互矛盾的,而在不同状态中,在相关条件未改变情况下,重复发出同样操作,则为冗余操作,应予以摒弃而只保留1个。

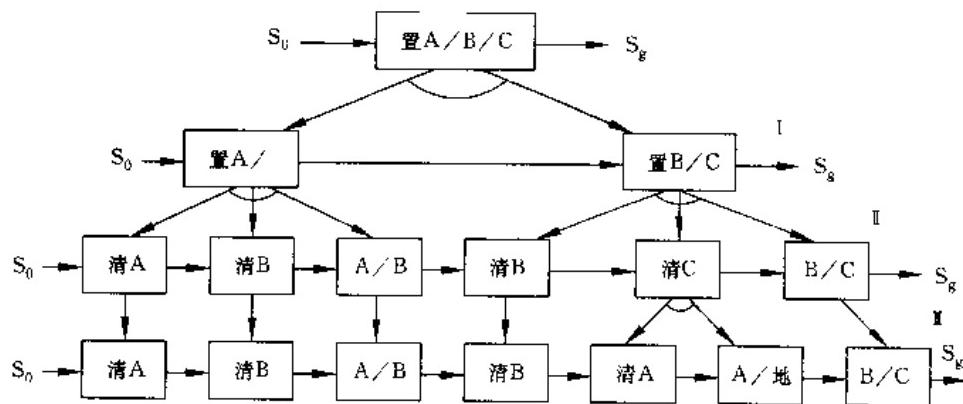


图7-13 机器人搬积木问题原始三级规划

图7-14为消除了矛盾和冗余的已修正的三级规划。修正的方法可依照原问题约定条件,

逐级逐块进行检查,尽量先对底层大木块进行操作,再按序往上堆放,出现矛盾即协调,冗余的清除相对容易解决。

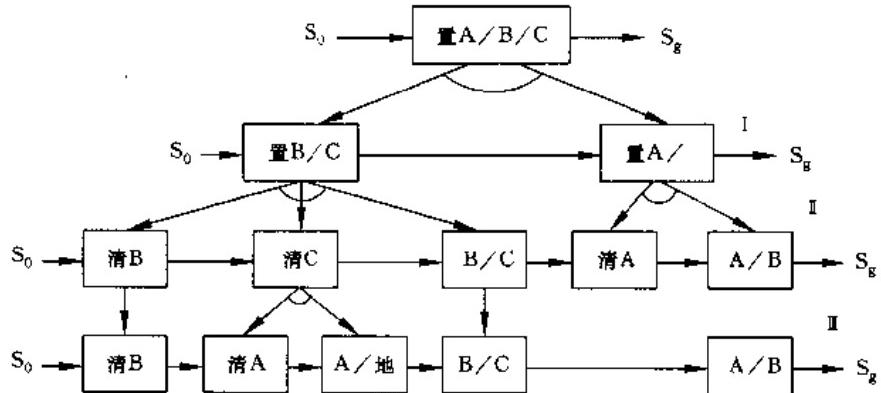


图 7-14 搬积木问题修正后的三级规划

## 习 题 七

1. 什么是人工智能规划求解系统? 它具有哪些功能和特点?
2. 考虑如图 7-15 所示的寻找路径问题的规划

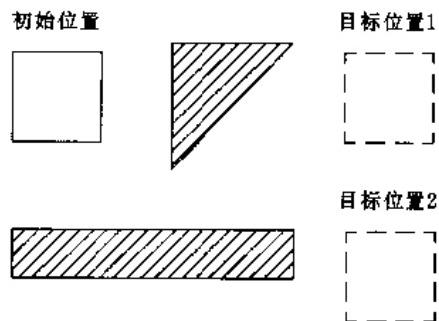


图 7-15 寻找路径问题的规划

(1) 对如图所示物体 A 和障碍物(阴影部分)建立一个结构空间。其中目标位置具有两种情况。

(2) 请寻求一个可将物体 A 移置到目标位置 1 及目标位置 2 的无碰撞移动的过程规划。要求问题只限于二维空间的平移及必要的旋转操作。

3. 请按条件匹配选择操作的方法(又称为 STRIPS 规划技术),设计一个清扫厨房的过程规划。设计中要注意下述约束情况:

- 在清洗地板之前,必须先行打扫。
- 在打扫地板之前,必须先把垃圾桶拿出去。
- 清扫火炉或电冰箱会弄脏地板。
- 清扫电冰箱会造成垃圾污物,并把工作台弄脏。
- 清洗工作台或地板将弄脏洗涤盘。

4. 小机器人 Piver 想进入房内与朋友下棋,但不能开门让自己去,而只能不停地喊叫,让叫声促使开门。机器人 Maxen 在房内,他喜欢平静,不喜欢人多,也不喜欢吵闹。Maxen 可以

开门使 Piver 停止叫喊,但随 Piver 还有 Masa 和 Robin 要进入。Masa 喜欢 Robin,但不会下棋,Robin 喜欢玩各种游戏。Piver 喜欢下棋,还愿意陪 Masa 玩。房间内有一只风筝,可由 Maxen 或 Piver 送给其中任何人。试猜想 Maxen 和 Piver 各能采取什么规划以期得到理想的环境。请写出该规划过程的描述。

5. 用本章学过的任何一种规划系统生成方法,解决图7-16的机械手(或机器人)堆积木的问题。

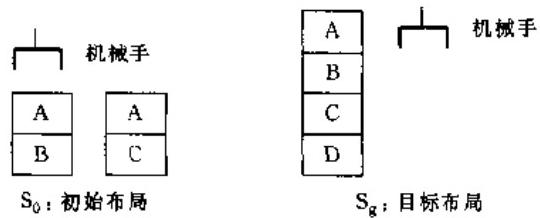


图7-16 堆积木规划问题

6. 针对 lanoil 塔问题,规划一个3阶问题的搜索式解过程。

## 第八章 机器学习

在第一章已论述过系统是否具有学习能力是衡量该系统是否有智能的一个显著标志。学习能力被认为是系统智能行为所具有的最基本属性。同时，机器学习(machine learning)也是使计算机具有智能的根本途径。正如桑克(R·Shank)所说，“一台计算机若不会学习，就不能说它具有智能。”

在这一章，首先讨论有关机器学习的一些基本概念，随后论述机器学习的分类情况和它的发展历史，最后通过实例分析机器学习的机制。

### 第一节 机器学习的概念

#### 一、什么是学习

关于什么是机器学习的问题，至今仍没有严格定义，能够找到的比较满意的说法是西蒙的定义。他说：“学习是使系统做一些适应性的变化，使得系统在下次完成类似的任务时比前一次更有效。”

另一种观点认为学习就是知识的获取。的确，知识获取似乎是大多数机器学习系统的中心任务，但是，所获取的知识有时并不能使系统得到改善。另外，有些学习过程并不能用获取的知识来描述，因为这些活动过程主要不是思维活动，而是通过反复练习来实现的（如小孩学走路、大人学骑自行车等）。

第三种观点以米查斯基(R·S·Michalski)为代表，他认为“学习是构造或修改所经历事物的表示。”人们知道，系统为了获得知识，就必须采用某种形式对知识进行表示和存储。按照这种观点，学习的核心问题是构造客观现实的表示，而不是对系统性能的改善，性能的改善仅仅是构造表示的效应。

综合上述三种观点可以得出，学习是对某一个特定目标的知识获取的智能过程，系统的内部表现为新知识结构的建立和改进，外部表现为系统性能的改善，变得更快、更精确、更健全。

#### 二、人类学习与机器学习

研究机器学习有两个基本目的。其一是想理解学习本身，通过研制计算机学习模型，获取对人类学习方法的了解；其二是为计算机提供学习的能力。研制能够受教育，而不只是接受程序的计算机系统，早就是人工智能的一个研究目标。基于这种考虑，有必要对人类学习与机器学习加以分析比较。

##### （一）人类学习是一个漫长的过程

学习的缓慢性是人类学习的最显著特征。一个人从呱呱坠地就开始了学习过程。从学习说

话、走路到上小学、中学一直到大学，学习文化知识和科学技术，要花掉二十几年的时间，才能成为某个领域、某个专业的行家里手。这说明人类是非常缓慢的学习者。

机器学习可以比人类学习快得多。这是由于机器学习可以将人类学习中的无效成分去掉，而保留以后能使用的有益模式和有用信息。因此，避免人类学习的缓慢性成了机器学习研究的一个目标。

### (二) 人类学习不存在复制过程

不能复制是人类学习的另一个特征。要是能将一个诺贝尔奖获得者所具有的知识复制给另外一个人，那将是多么诱人！然而，这是不可能的。这可能又回到所谓的“生物大脑复制”的问题上来了，我们认为生物大脑的复制并不是人工智能想要达到的目标。与此相反，复制工作对于计算机来说是最为方便的。

### (三) 人类学习可能会遗忘

人类在漫长的学习活动期间，可能会忘记曾经所学习过的东西，也正是这种学习的遗忘特征，使得人类能够从记忆中删除那些冗余、矛盾的知识，不断地完善自己的知识。

然而，通过学习获取并保存在计算机中的知识，可以永久保存下去。除非人为地删除。

### (四) 人类学习是逐渐积累的过程

无论是从某个特定个体人或是从整体人类角度分析，人类所掌握的各种知识和技能是通过不断学习和积累进行的，这种学习方式是一种动态的学习方式。

与人类不同，机器学习的知识素材都是由人给予组织的，知识之间的联系基本上是固定不变的，存储在机器中的知识往往很少，自动获取知识的能力也非常有限，远没有达到人脑所积累知识的水平。

分析人类学习和机器学习差异的目的在于把人类学习的特长，融合在机器学习过程中，研究出高水平的机器学习模型。

## 第二节 机器学习系统

为使机器具有某种程度的学习能力，使之通过学习增长知识，改善性能，提高系统的智能水平，需要建立相应的学习系统。

### 一、什么是机器学习系统

机器学习系统是能在一定程度上实现机器学习的软件。Saris 在1973年对机器学习系统定义为：如果一个系统能够从某个过程或环境的未知特征中学习到有关信息，并且能把学到的信息用于对未来的估计、分类、决策或控制，以便改进系统的性能，那么它就是一个学习系统。Smith 等人在1977年也给机器学习系统作了类似的定义。他们说：如果一个系统在与环境相互作用时，能利用过去与环境作用时得到的信息，并提高系统的性能，那么这样的系统就是机器学习系统。

上述定义表明，一个机器学习系统应具有以下功能：

### (一)具有适当的学习环境

学习系统中环境并非指通常的物理条件,而是指学习系统进行学习时所必需的信息来源。

### (二)具有一定的学习能力

一个好的学习方法和一定的学习能力是取得理想的学习效果的重要手段。所以,学习系统应模拟人的学习过程,使系统通过与环境反复多次相互作用,逐步学到有关知识,并且要使系统在学习过程中通过实践验证、评价所学知识的正确性。

### (三)能用所学的知识解决问题

学习的目的在于应用,正像Saris定义中所说的,学习系统能把学到的信息用于对未来的估计、分类、决策和控制。

### (四)能提高系统的性能

提高系统性能是学习系统的最终目标。通过学习,系统随之增长知识,提高解决问题的能力,使之能完成原来不能完成的任务,或者比原来做得更好。

## 二、一种机器学习系统模型

由上一小节的分析,可以看出一个学习系统至少应有环境、知识库、学习环节和执行环节四个基本部分。一种典型的机器学习系统模型如图8-1所示。下面介绍其主要组成部分的功能。

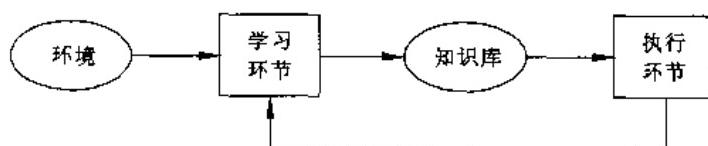


图8-1 一种机器学习系统模型

### (一)环境

系统中的环境包括工作对象和外界条件。比如在医疗系统中,环境就是病人当前的症状、物化检验的报告和病历等信息;在模式识别中,环境就是待识别的图形或景物;在控制系统中,环境就是受控的设备或生产流程。

环境提供给系统的信息水平和质量对于学习系统有很大的影响。信息的水平是指信息的一般性程度,也就是适用范围的广泛性。高水平的信息往往比较抽象,适用面更广泛,而低水平的信息往往比较具体,只适用于个别问题。

信息的质量指信息的正确性、信息选择的适宜性和信息组织的合理性。信息质量对学习难度有明显的影响。比如,如果示教者向系统提供准确的示教例子,而且提供例子的次序也有利于学习,那么系统就易于归纳;否则示教例子中有干扰,或者示教例子的次序不合理,则系统就难以进行归纳。

## (二) 学习环节

学习环节是系统的学习机构,是学习系统的核心。它通过对环境的搜索取得外部信息,然后经分析、综合、类比、推理等思维过程获得知识,并将这些知识送入知识库中,供执行环节使用。

事实上,由于环境提供的信息水平与执行环节所需的信息水平之间往往有差距,学习环节的任务就是解决这个水平差距问题,如果环境提供较高水平的信息,学习环节就去补充遗漏的细节,以便执行环节能用于具体情况。如果环境提供较具体的低水平信息,即是在特殊情况执行任务的实例,学习环节就要由此归纳出规则,以便系统能完成更为一般的任务。

## (三) 知识库

学习系统设计的另一个重要问题就是知识库的形成设计以及其内容。

知识库的形式就是知识表示的形式,选择知识表示方法要考虑下面几个准则。

### 1. 可表达性

特征向量表示知识适合于描述缺乏内在结构的事物,因为它是用一个固定的特征集合来描述的。谓词逻辑则可以适合于描述结构化的事物。

### 2. 推理难度

学习系统中一种常用的推理是比较两个描述是否等效。显然,判定两个特征向量的等效较容易,而判定两个谓词表达式等效较难。

### 3. 可修改性

特征向量和谓词逻辑这类显式的知识表示都容易修改;而像过程表示这类隐式的知识表示方法就难以修改。

### 4. 可扩充性

可扩充性是指学习系统通过增加词典条目和表示结构来扩大表示能力,以便学习更加复杂的知识。学习系统实质上就是对原有知识库的扩充和完善。

## (四) 执行环节

执行环节实际上是由执行环节和评价环节两部分组成,执行环节用于处理系统面临的现实问题,比如定理证明、智能控制、自然语言处理、机器人行动规划等;评价环节用来验证、评价执行环节执行的效果,比如结果的正确性等。评价环节的处理方法有两种,一种是把评价时所需的性能指标直接建立在系统中,由系统对执行环节所做出的结论进行评价;另一种是由人来协助完成评价工作。

另外,从执行环节到学习环节必须要有反馈信息。这样,学习环节就可以根据反馈信息决定是否要从环境中获取进一步的信息进行再学习,以便修改、完善知识库中的知识。

# 第三节 机器学习分类

当前国际上流行的机器学习分类方法主要有四种:按应用领域分类,有专家系统、问题求解、认识模型等;按获取的知识的表示分类,有谓词逻辑表达、产生式规则、决策树、框架、神经网络等;按推理策略分类,如演绎推理和归纳推理。按学习系统性分类的方法综合考虑了事物

的历史渊源、知识表示、推理策略和应用领域等因素，是对前面三种分类方法的综合。下面按推理策略分类介绍机器学习的诸方法，然后扼要地介绍一下按系统性分类的情况。

## 一、基于推理策略的分类

一个学习过程实质上是学习系统把环境所提供的信息转换成新的形式，以便存储和使用。这种信息的转换就是一种推理，而推理的性质确定了学习策略的类型，在学习过程中，学生所使用的推理越少，他对教师的依赖就越大，因而教师的负担就越重，反过来，学生使用的推理越多，教师的负担就越轻。显然，基于推理策略分类方法可以按学生使用推理的多少和难易程度进行。下面分别进行讨论。

### (一) 机械学习(Rote Learning)

机械学习是最简单的学习方法，它亦被称为记忆学习或死记硬背式学习。这种学习方法不需要推理，而是由教师向系统提供被记忆的信息，并用这些信息指导系统的行为。

Samuel 的西洋跳棋程序就是机械学习的典型例子。该程序学习下跳棋学得相当好，以至它可以击败它的研究者。Samuel 的程序采用极大极小搜索过程去搜索博奕树，像其他程序一样，系统只限定它搜索树的若干层（确切的层数视具体情况而定）。当不能搜索更远时，它将用静态估价函数对棋局进行评估，再以所评估值去继续对该博奕树搜索。当完成树的搜索后，往回倒推评分值，并得出表示树根位置的评分，然后选择最好的走步并执行，而且在树的根结点处记录下棋局的状态和刚计算出来的倒推评分，如图 8-2(a) 所示。

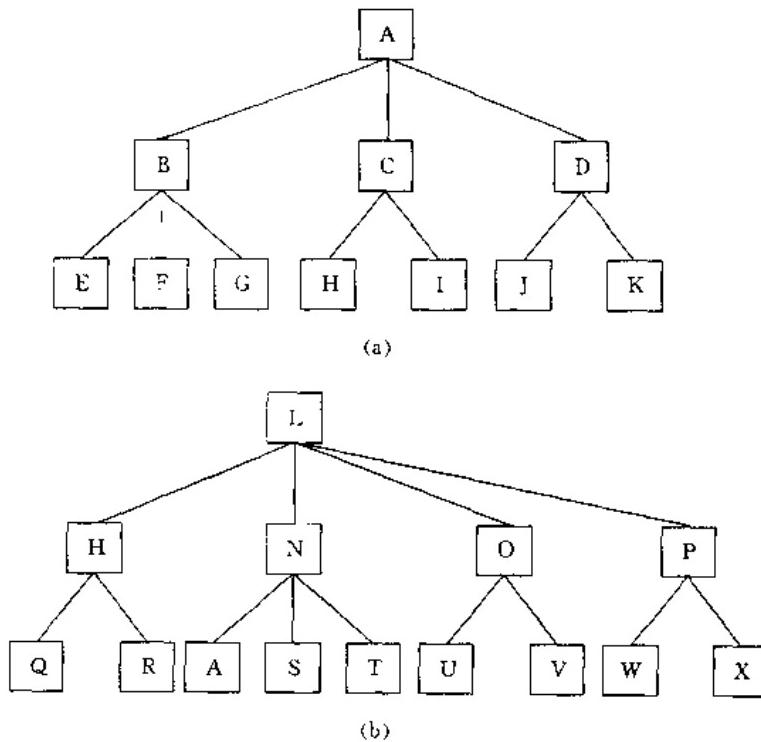


图 8-2 存储倒推而上的值

假若在以后的博奕中，出现了如图 8-2(b) 所示的情况，处于 A 的评分可用已存储的值；而

不必用静态估价函数去计算。因为关于  $A$  的值恰好是从这种搜索中倒推而上所计算出的值，其效果相当于已附带搜索了若干层。

机械学习是记忆，它仅保存新的知识以便使用。这里是个检索问题，而不是重复计算、推理或查询。机械学习可以认为是基本的学习方式，它本身并不能实现智能学习，但是它是其他学习系统所固有的重要组成部分。在机械学习系统中，知识已经以某种方式获取，并且是一种直接可使用的形式。其他更高级的学习系统首先从实例或者从指导获取知识，然后再记忆。因而所有学习系统都是建立在机械学习的基础之上，即对知识库中的知识进行存储、维护和检索。

对于机械学习系统有下面三个问题是值得考虑的：

#### 1. 存储组织问题

在系统中，只有当检索所耗时间小于重新计算所耗时间时，机械学习才是有用的。这就要求系统设计时适当地组织存储结构，使得检索速度加快。

#### 2. 环境稳定性问题

环境的迅速变化会直接影响系统中知识的有效性，过时的信息显然会影响其适用性。解决这一问题的方法是：其一是机械学习系统必须能够监控环境的每一种变化，使得系统始终保持存储最新的信息，其二是检查被检索的信息是否仍归有效，采取的方法是在存储信息的同时，还存储一些反映当时环境的辅助数据，当进行信息检索时，系统中存储的环境与当前状态进行比较，再来决定该信息是否仍然有效。

#### 3. 存储与计算的权衡问题

如果要记忆信息的存储和检索所消耗的代价超过了重新计算的代价，这种现象是不能允许的，因而要对存储与计算两者进行权衡。权衡的基本方法有两种：其一是当信息第一次使用时，就要决定是否将它存储起来供以后使用。这时要对信息存储所需空间和将来进行重新计算的代价进行权衡；其二是预先存储信息，以后再来决定是否将它遗忘。这样使得系统始终保存着最需要的信息。

## (二) 讲授学习(Learning from Instruction)

讲授学习(指导学习)系统中，由外部给系统提供抽象的、一般化的信息，学习系统经过选择和改造，把新的信息与系统原有的知识融为一体。由于外部提供的信息过于抽象，它的水平高于执行时所用信息的水平，因此学习环节要把较高水平的知识转换为较低水平的知识，这种转换称为实用化。

研究讲授学习的途径主要有两种。一是在开发系统时接收抽象的、高级的信息，并将它转换成规则，以便指导执行部分。二是开发完善的工具，比如知识库的编辑和调试辅助程序，使得专家们可以很方便地将专门知识转换成详细的规则。

专家知识转换成程序一般由咨询、解释、实用化、归并和评价五个部分组成。

1. 咨询 程序的第一步是要对专家咨询。咨询工作可以只是请求专家给出一般的意見，也可以要求专家评价知识库中已有的知识以修改它们的策略。

2. 解释 接受专家的建议并转换成内部表达形式。如果专家建议以自然语言形式给出，就有一个自然语言理解的问题，因此解释过程就相当麻烦。为了简化解释过程，专家的建议应以受限制自然语言给出。

3. 实用化 专家意见被接收并解释成明确的形式表达后，仍然不能为执行部件直接执行。所以要进一步对其进行实用化处理。也就是将抽象的专家建议转化为可以执行的规则。

4. 归并 在把知识加入到知识库的过程中,可能产生两个问题。其一是知识的重叠应用;其二是产生矛盾性动作。解决的办法是对新产生的规则进行修改,使之不再发生应用的重叠和矛盾。另外一种办法是利用原规则排序,以确定哪些规则应优先应用,用来解决矛盾性动作问题。

5. 评价 对经过解释、实用化、归并的专家知识进行评价。并将系统的评价反馈到咨询步骤,通知专家并征求修改意见。

### (三) 演绎学习(Deductive Learning)

这种学习方法是一种常规的逻辑推理方法。其推理的过程通常从公理出发,经过逻辑变换,推导出结论。演绎学习包括知识改造、知识编译、生成宏操作、保持等价操作和其他的一些保真变换。

与逻辑推理不同,归纳推理(Inductive Inference)是另外一种推理过程。归纳推理能够对输入信息进行推广(generalization)并且选择其中较理想的结果。与逻辑推理比较,归纳推理不是保真变换,而是“保假”变换,即若前提是假,那么归纳出的结论也是假的。比如命题:鸟会飞,由保假性可以得知,若 A 不是鸟,那么 A 就不会飞。归纳推理是人类最重要的一种思维方式,它也是发现科学定律和定理的思想武器,归纳推理和演绎推理互为逆过程,大多数学习系统中它们同时出现。

### (四) 解释学习(Explanation-based Learning)

解释学习是近年来出现的一种机器学习方法。它利用问题求解的示例,依赖领域知识构造出求解过程的因果解释结构,并获取控制知识,为以后类似问题求解提供指导。

解释学习过程可分成两个步骤:

#### 1. 产生解释

输入实例后,系统首先对问题进行求解。比如由目标引导反向推理,从领域知识库中寻找有关规则,使其后件与目标匹配。找到这样的规则后,就把目标作为后件,该规则作为前件,并记录这一因果关系。然后以规则的前件作为子目标,进一步分解推理。如此反复沿因果链进行,直到求解结束。一旦得到解,便证明了该例的目标可满足,同时获得了证明的因果解释结构。

构造解释结构通常有两种方式:其一是将问题求解每一步推理所用的算子汇集,构成一个动作序列作为解释结构;其二是自顶向下的遍历证明树结构。

#### 2. 对得到的解释结构和事例进行概括

概括通常采取的方法是将常量转换成变量,且去掉某些不重要的信息,仅仅保留在求解时所必需的那些关键信息,经过一定的方式进行组合形成产生式规则,从而获得概括性的控制知识。

著名的解释学习系统有 G·DeJong 的 GENESIS, T·Mitchell 的 LEX 和 LEAP 以及 S·Minton 的 PRODIGY。

### (五) 类比学习(Learning by Analogy)

类比学习是演绎学习和归纳学习的组合。它对不同论域的描述进行匹配,确定公共的子结构,以此作为类比映射的基础。寻找公共子结构是归纳推理,而实现类比映射是演绎推理。

假设有两个不同的论域:源域和目标域,并且已知这两个论域具有相同或相似的性质,那

么利用类比可以猜测源域中其他性质也可能在目标域中存在。下面给出形式化的说明。

设  $S$  和  $G$  分别表示源域和目标域,  $S$  中的元素  $s$  同  $G$  中的元素  $g$  相对应, 并具有相同的性质  $P$ , 即  $P(s)$  和  $P(g)$  成立。若  $S$  还具有性质  $Q$ , 即  $Q(s)$ , 则可以推出  $g$  也有性质  $Q$ , 即  $Q(g)$  成立。亦即

$$P(s) \wedge Q(s), \quad P(g) \vdash Q(g)$$

类比推理过程分成两个步骤。第一步是找出源域和目标域的公共性质  $P$ ,  $P$  同源域另一性质的关系  $P(s) \rightarrow Q(s)$ , 并且推广到  $\forall x(P(x) \rightarrow Q(x))$ ; 第二步是从源域到目标域的映射以得到目标域的新性质, 即从  $\forall x(P(x) \rightarrow Q(x))$  和  $P(g)$  应用假言推理, 可得到  $Q(g)$ 。

温斯顿(P·Winston)曾描述了一个以框架作为知识表达的类比学习系统。在此系统中, 类比学习就是把一个框架(源)中的槽值传到另一个框架(目标)的槽中, 这种传送分成两步进行:

- (1) 利用源框架产生所推荐的槽, 这些槽的值可传送到目标框架;
- (2) 利用目标框架中已有的信息来过滤由第一步所推荐的相似性。

## (六) 归纳学习(Inductive Learning)

归纳学习是教师或环境提供某种概念的一些例子, 学生利用归纳推理将这些例子推广, 产生该概念的一般描述。归纳学习是最基本的, 发展较成熟的学习方法。归纳学习可分为实例学习和观察与发现学习。

### 1. 实例学习(Learning from Examples)

实例学习也称为概念获取(Concept Acquisition)。是由教师提供给系统某种概念的正例集合和反例集合, 学习通过归纳推理产生覆盖所有正例并排除所有反例的该概念的一般描述。这些正例是由已知该概念的教师或者是学生做实验时从系统中得到的反馈信息而提供的。

实例学习是人类最基本的学习方式, 因而也是机器学习的主要研究对象。同时, 实例学习可用于专家系统知识库的自动构造, 所以它已成为机器学习的核心研究领域。

当前对实例学习的研究集中在两个方面, 即例子—类型的一般化和部分—整体的一般化。

在例子—类型一般化中, 给系统提供某一类对象的独立实例, 系统的目标是归纳出这些类的一般描述。对象可以是结构化的部件、几何形状、疾病描述、故事、问题的解、控制算子等。在部分—整体的一般化中, 任务是假设整个对象(情景、情况、过程)的描述, 但只给系统对象的局部。例如, 只给定一个房间的几张局部的照片, 要重构房间的整个视图。又比如只看到一个序列或过程的一部分, 要确定描述该序列或过程的规划。

实例学习的系统较多, 其中, 较有影响的有 E·B·Hunt 等的 CLS, J·R·Quinlan 的 ID<sub>3</sub>, Michalskyi 的 AQ<sub>11</sub> 等。

### 2. 观察与发现学习(Learning from Observation and Discovery)

观察与发现学习是由环境提供一组观察事例, 学生构造一个一般的概念描述(即理论)来覆盖所有或大多数事例。这是一种无导师学习。这类学习又分为观察学习与机器发现两类。

(1) 观察学习 观察学习是学生将已知事例进行分类, 同时产生每一类的—般概念描述。观察学习又可根据是否渐近(incremental)方式而分为概念形成和概念聚类。

#### a. 概念聚类(Conceptual Clustering)

概念聚类的思想是 R·S·Michalskyi 于 1980 年提出的。该方法对已知事例分类, 首先产生每一类概念的描述, 然后再用该概念描述指导下一步的分类, 这样循环做下去直到得到满意的结果为止。这类著名的系统有 R·S·Michalskyi 和 R·E·Stepp 的 CLUSTER/2 和 P·Langley 等

的 GLAUBER。

b. 概念形成(Concept Formation)

概念形成是学生对依次出现的事例进行分类,产生概念描述,同时构造聚类的层次结构。著名的概念形成的系统有 E·A·Fogelbom 的 EPAM、M·Lebowitz 的 UNIMEM 和 D·F·Sher 的 COBWEB。

(2)机器发现(Machine Discovery)

学生从观察的事例或经验数据中进行归纳产生规律或规则,这就是机器发现。机器发现是观察与发现学习的最困难、最富有创造性的一种学习形式。机器发现包括有经验发现和知识发现两种类型。

a. 经验发现(Empirical Discovery)

经验发现是指从经验数据中发现规律和定律。著名的经验发现系统有 Langley 的 BA-CON. 4 和 B·Falkenhainer 与 Michalsky 的 ABACUS 等。

b. 知识发现(Knowledge Discovery)

知识发现是从已知观察事例或数据中发现知识(一般是产生式规则)。这是一个新兴研究的领域,有关知识发现的系统有 Langley 的 GLAUBER、J·G·Ganascia 的 CHARADE、R·M·Goodman 与 P·Smyth 的 ITRVLE 等。

基于推理策略的分类与机器学习的发展历史比较吻合。但是,随着研究的纵深发展,其研究的领域不断扩大,原有的学习方法更加成熟,新的学习方法接踵而来。如联接学习的兴起、遗传算法脱颖而出、解释学习变成了热门……凡此种种使得以往的分类方法已无法兼顾这些新的领域。为了适应这些新的情况,近年来提出了按系统性进行学习分类。

## 二、基于系统性的分类

基于系统性分类,机器学习可分为四种类型,即:归纳学习、分析学习、联接学习、遗传算法与分类器系统。

1. 归纳学习

前面已经论述过,这里不再讨论。

2. 分析学习(Analytic Learning)

分析学习是针对几个实际例子,应用领域知识进行分析来学习。分析学习的主要特征有:

(1) 推理策略是演绎而不是归纳;

(2) 使用过去问题求解的经验,指导新问题求解,或产生能更有效地应用领域知识的搜索控制规则。也就是说,分析学习的目标是改善系统效率,而不是扩充概念描述的范围。分析学习主要包括解释学习、类比学习、多级组块(chunking)、迭代的宏操作和实例学习(case based learning)等。

3. 联结学习

一个联结模型是由一些类似神经元的简单单元带权互连而组成的网络。在这样的网络中,每个神经元都具有某种状态,其状态由与该神经元相连的其他神经元输入而决定。联结学习通过使用各类不同的样本训练网络而实现。联结学习的目标是区分输入模式的等价类。网络通过对样本的训练产生网络的内部表示,用来识别其他的输入模式。学习过程主要是调整表现在网络中的联结权。联结学习是非符号的,并且具有高度并行分布式处理的能力,近年来受到了广

泛的关注并得到了实际的应用。

#### 4. 遗传算法(Genetic Algorithm)与分类器系统(Classifier Systems)

分类器系统是一种高度并行的信息传递(message-passing)的规则库系统。分类器系统通过信任分配(桶队列算法)和规则发现(遗传算法)来学习。遗传算法模拟生物繁殖的突变(互换、倒位、点突变等)和达尔文的自然选择(在每一生态环境中适者生存)。具体来说,一个概念描述的变形(分类器)对应于一个物种的个体,这些概念的诱导变化和重组用一个目标函数(相当于自然选择准则)来衡量,看其中哪些能够保留在基因库中。遗传算法适用于非常复杂困难的环境。比如,具有噪音和无关数据的不断更新的事物,不能明显和精确地定义的问题目标以及通过很长的执行过程才能确定当前行为的价值等。

### 第四节 机器学习的发展简史

从50年代以来,机器学习几乎是伴随着人工智能一起发展的。在各个不同时期中,机器学习的研究目标、研究方法、研究的倾向与侧重面各有所不同。根据这一历史现象,一般把机器学习划分为四个发展阶段。

#### 1. 神经元模型研究阶段

这个时期具有代表性的工作有 F·Rosenblatt 的感知机(1958年);N·Rashevsky 数学生物物理学(1948年);W·S·McCullouch 与 W·Pitts 的模拟神经元的理论(1943年);R·M·Friedberg 对进化过程的模拟等。

#### 2. 符号概念获取研究阶段

随着有影响的 P·Winston 定理的发表,人们对机器学习的兴趣得到了恢复,60年代初期,机器学习的研究进入了第二阶段。

在这个阶段,心理学和人类学习的模拟占有主导地位,其特点是使用符号而不是数值表示来研究学习问题,其目标是用学习来表达高级知识的符号描述。在这一观点的影响下,研究人员一方面深入探讨学习的简单概念,另一方面则把大量的领域知识并入学习系统,以便它们发现高深的概念。

这个阶段代表性的工作有 Hunt 和 C·I·Hovland 的 CLS 系统(1963年),P·Winston 的结构学习系统(1975年),B·G·Buchana 等人的 META-DENDRAL(1978年)知识获取系统和 Michalski 的 AQVAL(1973年)。

#### 3. 符号学习兴旺发达阶段

机器学习发展的第三个阶段始于70年代中期,这个阶段表现的符号学习的兴旺发达。当时由于专家系统正处于黄金时期,专家系统的研制成果不断涌现,使得知识获取成为当务之急,相应的有关学习方法相继推出。比如 D·J·Mostow 的讲授学习(1983年),Michalski 和 Steep、Langley 的观察与发现学习(1983年),Winston 的类比学习(1979年),DeJong 的解释学习(1983年)等。

#### 4. 联结学习和符号学习共同发展阶段

80年代后期以来,形成了联结学习和符号学习共同发展的局面,这个时期是机器学习的第四个阶段。在这个时期,发现了用隐单元来计算和学习非线性函数的方法,从而克服了早期神经元模型的局限性。同时,由于计算机硬件的迅速发展,使得神经网络的物理实现变成可能,在声音识别、图像处理等领域,神经网络取得了很大的成功。

在这个时期,符号学习伴随人工智能的进展也日益成熟,应用领域不断扩大,最杰出的工作有分析学习(特别是解释学习)、遗传算法、决策树归纳等。

通过对以上机器学习进展的回顾和分析,我们可以预计机器学习今后的发展方向将集中在以下几个方面:

- (1)加强机器学习形式理论的研究,建立起自身的理论体系;
- (2)进一步扩大实际应用的领域范围,在应用中发展和完善现有的方法;
- (3)为了解决更加复杂的任务和模拟人类思维过程,协同多种学习方法工作,构造集成化的学习系统将会引起人们的重视。

## 第五节 从例子中学习

从例子中学习属于归纳学习,是目前机器学习方法中最成熟的方法之一。

### 一、概述

从例子中学习要求环境能够从一些特殊的实例(这些实例事先由教师划分为正例和反例两类),并由这些实例进行归纳推理,导出一般性的规则。给系统提供的这些供学习的正例和反例所包含的是非常低级的信息,系统经学习环节可归纳出高水平的信息即规则,并且在一般情况下,可用这些规则指导执行环节的操作。

例如,在一个学习下棋的学习系统中,训练这个系统的一种方法是给它提供具体的棋局,并告诉它什么是最好的走步。系统必须一般化这些具体走步以便发现妙招的策略。再比如,我们想教给系统有关“狗”的概念,可以提供给系统各种动物和各种其他物体,说明每个物体的特点,并说明它们是不是狗。这就是狗的正例和反例,由此系统根据物体的特征,推出识别狗的规则。

### 二、从例子学习的两个空间模型

Simon 和 Lea 1974 年在一篇有关归纳的早期文章中,把从例子学习的问题描述为使用示教例子来指导对规则的搜索,并提出从例子学习的两个空间模型概念,即实例空间和规则空间。实例空间是所有示教例子的集合,而规则空间是所有规则的集合。他们的基本观点是:从例子学习系统应在规则空间中搜索所求的规则,并在实例空间中选择一些示教例子,以便解决规则空间中某些规则的歧义性。系统就是这样在实例空间和规则空间中交替进行搜索,直到找到所

要求的规则。图 8-3 表示从例子学习的两个空间模型。

在图 8-3 中,除描绘从例子学习的实例空间和规则空间外,还描绘了解释实例和实验规划的过程。在这个模型中,首先由示教者给实例空间提供一些初始示教例子,然后程序对示教例子进行解释。由于示教例子的形式往往不同于规则形式,所以有必要对例子进行解释。

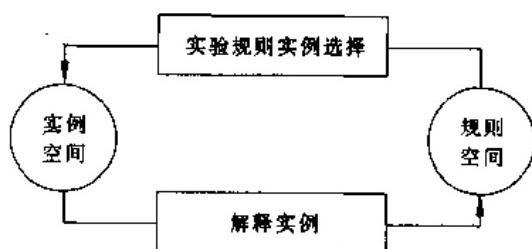


图 8-3 从例子学习两空间模型

往后再利用被解释的示教例子去搜索规则空间。一般情况下不能一次就从规则空间中搜索到要求的规则，因此还要寻找一些新的示教例子，这个过程就是选择例子。此过程如此循环，直到搜索到要求的规则。

作为两空间模型的例子，下面考虑教计算机程序扑克牌中“同花”概念的问题。同花是指五张同一花色所组成的一手牌。在这个学习问题中，实例空间是五张牌的全部各手牌的集合。我们可以把这个空间中单个的点表示为一组五个有序对，比如：

$\{(2, \text{梅花}), (3, \text{梅花}), (5, \text{梅花}), (\text{J}, \text{梅花}), (\text{K}, \text{梅花})\}$

每一有序对指明一张牌的点数和花色。整个实例空间是所有这样的五张牌集合的空间。

这个问题的规则空间是描述一手牌的全部谓词表达式的集合。这些表达式使用下列符号描述：谓词 SUIT(花色)和 RANK(点数)；表示牌的变量  $c_1, c_2, c_3, c_4$  和  $c_5$ ；某些必要的自由变量；常量梅花，方块，红心，黑桃，A, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q 和 K；合取联结词  $\wedge$ ，以及存在量词  $\exists$ 。比如同花的概念可以表示为下列规则：

$$\exists (c_1, c_2, c_3, c_4, c_5) : \text{SUIT}(c_1, x) \wedge \text{SUIT}(c_2, x) \wedge \text{SUIT}(c_3, x) \wedge \text{SUIT}(c_4, x) \wedge \text{SUIT}(c_5, x)$$

采用两空间方法的学习系统利用了封闭世界假设，即规则空间包含着所要求的概念。封闭世界假设使得程序能通过不断排除已知不正确的候选概念，找到所要求的概念。下面，我们围绕两空间模型的论述，依次介绍实例空间、解释例子、规则空间和实验规则等四个部分。

### (一) 实例空间

实例空间涉及到两个问题：示教例子的质量和实例空间的搜索方法。

#### 1. 示教例子的质量

高质量的示教例子是无二义性的，它可以为规则空间的搜索提供可靠的指导。低质量的示教例子会引起互相矛盾的解释，其结果仅为规则空间的搜索提供试探性的指导。

例如，在教给程序“同花”概念时，可能会产生几种二义性。它们使得程序难于发现正确的概念。首先，实例可能有错误。如果实例的描述不正确，会产生测量错误。比如梅花2被错认为黑桃2。另一类错误是分类错误，比如把一手非同花的牌误判为同花，或反之。

另外，示教例子排列次序也会影响学习的质量。如果示教者对示教的例子事先按特征的重要性进行合理的排序，学习任务就比较容易些。

#### 2. 实例空间的搜索方法

在人工智能的研究工作中，不很注意这个问题，因为一般情况下认为实例是同时提供的，即程序并不选择实例。之所以这样做，是因为程序无法控制实例的选择。但是也有的程序可以主动地选择另外一些附加的实例，以便修正假设，这种方法称为补充学习。还有的程序直接搜索实例空间，这种方法称为主动选择例子。

搜索实例空间的目的在于选择合适的例子，使之能证实或否定规则空间中假设规则的集合  $H$ 。第一种方法是选择对划分规则空间最有利的实例，以便迅速地缩小在规则空间中的搜索区域。如果选定的示教例子把假设规则集  $H$  分成两半，那么在得到这个新实例后就可以排除一半的假设。第二种方法是在假设规则集  $H$  中选择最有希望的假设。第三种方法是利用  $H$  中的假设滤掉与那些期望为真的实例，以便学习程序能把注意力集中在违反当前假设的那些实例上。这种方法称为基于期望的过滤法。第四种方法是选择新的实例，使得计算费用为最少。

## (二)解释例子

解释示教例子的基本目的是提取指导规则空间搜索有用的信息。通常是把示教例子转换成易于进行符号归纳的形式。不过，这种转换也许是困难的，尤其是在感性的学习中。

例如，在 Winston(1970年)学习积木世界中“拱”概念的程序中，示教例子提供给程序积木世界结构场景的轮廓图，并说明该场景是拱的正例还是反例。解释例子的过程就是用积木世界知识来解释这个轮廓图，并用一些谓词和关系来描述这个轮廓图。比如，积木 A 是长方体，积木 B 是圆柱体，积木 C 是正方体。积木 A 放在积木 B 上面，积木 B 与积木 C 不接触等。这时解释例子的过程实际上就把轮廓图转换成为谓词和关系的表示形式，这样就便于归纳推理了。

## (三)规则空间

定义规则空间的目的是指定表示规则的操作符和术语。所谓规则空间是用指定的描述语言可以表示的所有规则的集合。与规则空间相关的两个问题是规则空间的要求和规则空间的搜索方法。而规则空间的表示方法又应支持归纳推理，所以先介绍归纳推理方法是必要的。

### 1. 归纳推理方法

归纳推理是由特殊到一般的推理。例如有规则“湖南人都很好客”，可以归纳推理为规则“中国人都很好客”。这样推理后的规则比原规则更普遍，但这种推理不一定保真。下面介绍几种归纳推理方法。

#### (1) 把常量变成变量

如学习扑克牌同花的概念，提供下面两个正例：

例1  $SUIT(c_1, \text{clubs}) \wedge SUIT(c_2, \text{clubs}) \wedge SUIT(c_3, \text{clubs}) \wedge SUIT(c_4, \text{clubs}) \wedge SUIT(c_5, \text{clubs}) \Rightarrow FLUSH(c_1, c_2, c_3, c_4, c_5)$

例2  $SUIT(c_1, \text{spades}) \wedge SUIT(c_2, \text{spades}) \wedge SUIT(c_3, \text{spades}) \wedge SUIT(c_4, \text{spades}) \wedge SUIT(c_5, \text{spades}) \Rightarrow FLUSH(c_1, c_2, c_3, c_4, c_5)$

其中例1表示五张梅花牌是同花，例2表示五张黑桃牌是同花，谓词 FLUSH 表示同花。现在把常量 clubs 和 spades 换成变量  $x$ （表示任意花色），就归纳出一条假设的规则。

规则1  $SUIT(c_1, x) \wedge SUIT(c_2, x) \wedge SUIT(c_3, x) \wedge SUIT(c_4, x) \wedge SUIT(c_5, x) \Rightarrow FLUSH(c_1, c_2, c_3, c_4, c_5)$

#### (2) 省略条件

为了学习同花概念，提供下列正例。

例3  $SUIT(c_1, \text{clubs}) \wedge RANK(c_1, 3) \wedge SUIT(c_2, \text{clubs}) \wedge RANK(c_2, 5) \wedge SUIT(c_3, \text{clubs}) \wedge RANK(c_3, 7) \wedge SUIT(c_4, \text{clubs}) \wedge RANK(c_4, 10) \wedge SUIT(c_5, \text{clubs}) \wedge RANK(c_5, k) \Rightarrow FLUSH(c_1, c_2, c_3, c_4, c_5)$

为了得到上述规则1，不仅要把常量 clubs 化为变量  $x$ ，而且要省略所有的 RANK 谓词。这说明一个概念的描述，可以简单地通过去掉合取项而得到归纳推理。

#### (3) 增加选择项

例如要程序学习“人面牌”的概念，人面牌是一张点数为 J、Q、K 的牌，提供下面的两个正

例。

例4  $\text{RANK}(c_1, J) \Rightarrow \text{FACE}(c_1)$

例5  $\text{RANK}(c_1, K) \Rightarrow \text{FACE}(c_1)$

将这两个正例合并得到等价式

$\text{RANK}(c_1, J) \vee \text{RANK}(c_1, K) \Rightarrow \text{FACE}(c_1)$

如果在此基础上,再增加一个析取项,就可以得到所要求的规则。

规则2  $\text{RANK}(c_1, J) \vee \text{RANK}(c_1, Q) \vee \text{RANK}(c_1, K) \Rightarrow \text{FACE}(c_1)$

这说明一个概念的描述可以通过增加析取项而得到归纳推理。

#### (4) 曲线拟合

曲线拟合适用于数值问题的归纳。例如,程序想发现系统的输出  $Z$  和两个输入  $x$  和  $y$  的关系。提供给程序  $(x, y, z)$  三元组形式的示教例子。

例6  $(0, 2, 7)$

例7  $(6, -1, 10)$

例8  $(-1, -5, -16)$

利用曲线拟合技术(如最小二乘回归法),程序将拟合出表示  $x, y, z$  之间关系的平面,即

规则4  $Z = 2x + 3y + 1$

#### 2. 对规则空间的要求

对规则空间有三个方面的要求,即规则的表示形式应适应归纳推理,规则的表示与实例的表示应一致,规则空间应包含要求的规则,下面分别进行分析。

##### (1) 规则的表示形式应适应归纳推理

上面介绍了四种归纳推理的方法。从中可以知道,不同的归纳推理方法要求不同的规则表示形式。比如把常量变成变量的方法要使用自由变量,增加选择项的方法要使用析取联结词等。前三种归纳推理方法要用谓词逻辑的方法表示规则,而第四种归纳推理方法要用状态矢量来表示实例,并用代数方程式来表示规则。

另外,在一定程度上,如果描述规则空间语言的表达能力越弱,那么可以使用的归纳方法就越少,规则空间的搜索范围也就越小,搜索就比较容易完成。但是由于这种规则空间包含的规则较少,它能解决的问题也就较少。因此,在设计系统时应该在规则空间的表达能力和规则空间搜索的难度之间进行权衡。

##### (2) 规则的表示与实例的表示一致

倘若示教例子和规则的表示形式相差很大,那么解释实例和选择例子的过程就很复杂。因此,最好采用相同的形式来表示规则和示教例子。比如要程序学习“对牌”的概念,对牌是两张点数相同的牌。对牌的规则是:

规则5  $\text{RANK}(c_1, x) \wedge \text{RANK}(c_2, x) \Rightarrow \text{PAIR}$

为了学习规则5,提供下面例9

例9  $(2, \text{clubs}), (3, \text{diamonds}), (2, \text{hearts}), (6, \text{spades}), (\text{K}, \text{hearts}) \Rightarrow \text{PAIR}$

这样表示的示教例子与规则5表示形式差异很大,使得归纳比较困难。如果把例9改用谓词逻辑表示,则有:

例10  $\text{RANK}(c_1, 2) \wedge \text{SUIT}(c_1, \text{clubs})$   
 $\wedge \text{RANK}(c_2, 3) \wedge \text{SUIT}(c_2, \text{diamonds})$   
 $\wedge \text{RANK}(c_3, 2) \wedge \text{SUIT}(c_3, \text{hearts})$

$$\begin{aligned} & \wedge \text{RANK}(c_4, 6) \wedge \text{SUIT}(c_4, \text{spades}) \\ & \wedge \text{RANK}(c_5, K) \wedge \text{SUIT}(c_5, \text{hearts}) \Rightarrow \text{PAIR} \end{aligned}$$

这样改写了后,再归纳出规则5就比较简单了。我们可以首先在例10中去掉五个 SUIT 条件,然后再去掉  $c_2, c_4$  和  $C_5$  的 RANK 条件,最后把常量2变换为变量  $x$ ,就可以归纳成规则5了。在这里就不必要对实例进行解释了。

### (3) 规则空间中应包含要求的规则

在有些实例学习的问题中,所要求的规则并不在规则空间中,即规则空间的描述语言不能够表示要求的规则。解决的办法是要引入新的术语,即扩充规则空间的描述语言。例如,用 RANK 和 SUIT 谓词表示对牌和同花的概念,但它们不能描述“顺牌”的概念。顺牌是五张点数相连的牌,比如8,9,10,J,Q 就是五个相连的点数,为此,就要引入新的谓词 SUCC。于是定义谓词 SUCC 为:

$$\begin{aligned} & \text{SUCC}(2, 3) \vee \text{SUCC}(3, 4) \vee \dots \\ & \vee \text{SUCC}(10, J) \vee \text{SUCC}(J, Q) \vee \text{SUCC}(Q, K) \vee \text{SUCC}(K, A) \end{aligned}$$

用扩充的描述语言可以写出顺牌的规则为:

规则6  $\text{RANK}(c_1, R_1) \wedge \text{RANK}(c_2, R_2)$   
 $\wedge \text{RANK}(c_3, R_3) \wedge \text{RANK}(c_4, R_4)$   
 $\wedge \text{RANK}(c_5, R_5) \wedge \text{SUCC}(R_1, R_2)$   
 $\wedge \text{SUCC}(R_2, R_3) \wedge \text{SUCC}(R_3, R_4)$   
 $\wedge \text{SUCC}(R_4, R_5) \Rightarrow \text{CONT}(c_1, c_2, c_3, c_4, c_5)$

## 3. 搜索规则空间的方法

下面介绍搜索规则空间的几种方法。这些方法都具有一个假设规则的集合  $H$ ,不同的仅仅是对于  $H$  的改进,以便得到要求的规则。

### (1) 变型空间法

变型空间法(version-space method)是一种数据驱动方法(data-driven method)。这种方法对规则和实例都采用同一种表示形式。初始的假设规则集  $H$  包括满足第一个示教正例和全部规则。在得到下一个示教例子时,对集合  $H$  进行一般化或特殊化处理,最后使集合  $H$  收敛为仅含要求的规则。

### (2) 改进假设法

改进假设法(hypothesis-refinement method)也是一种数据驱动方法。这种方法表示规则和实例的形式不统一。程序根据例子选择一种操作,用该操作去改进假设规则集  $H$  中的规则。

### (3) 产生与测试法

产生与测试法(generate and test)是一种模型驱动方法(model-driven method)。这种方法针对示教例子反复产生和测试假设的规则。在产生假设规则时,使用基于模型的知识,以便只产生可能合理的假设。

### (4) 方案示例法

方案示例法(schema instantiation)也是一种模型驱动方法。该方法使用规则方案的集合来约束可能合理的规则的形式,其中最符合示教例子的规则方案被认为是最合理的规则。

数据驱动方法的优点是可以逐步接受示教例子,以渐近方式学习,特别是变型空间法,它很容易修改集合  $H$ ,不要求程序回溯就可以考虑新的实例。而模型驱动方法难以逐步学习,它是通过检查全部实例来测试假设。在使用新假设时,它必须回溯或重新搜索规则空间。因为原

来对假设的测试已不适用于新实例加入后的情况。

模型驱动方法的优点是抗干扰性良好。由于使用整个实例集合，程序就可以对假设进行统计测量。在用错误实例测试假设时，它不因一、二个错误实例而放弃正确的假设。而数据驱动方法用当前的实例去修改集合  $H$ ，因此一个错误实例就会造成集合  $H$  的混乱。解决的方法是每次用新实例去修改集合  $H$  时，只作较小的修改，这样可以减少错误实例带来的影响，当然这也同时使得学习的进程变慢了。

#### (四) 实验规则

一旦学习环节根据示教例子搜索规则空间，并产生可能合理的假设规则集合  $H$  后，程序就可能需要收集更多的训练实例加以测试和修改集合  $H$ 。当实例空间和规则空间是以不相同的方法表示时，就需要判断训练哪些实例和怎样才能获得它们，这是一个复杂的过程。比如，假定一个遗传学习程序要想发现 DNA 的哪一部分是最重要的，为了测试几个高级假设（即假设的规则）要安排复杂的试验。这些试验合成 DNA 的特殊成分，并把它插入到适当的细菌细胞中，以观察细胞的最后动作。

AM 是执行某些实验规划的人工智能学习程序的例子，在 AM 产生一个新概念后，它必须收集该概念的实例，以便于评价和修改。AM 用几种方法收集实例，用启发式的方法收集边缘实例就是其中的一种方法。

### 三、学习单个概念

单个概念的学习是提供给系统一个概念的若干正例和若干反例，系统由此构成规则空间，并可得到在这个规则空间中的一个概念。这个概念应包括所有的正例，但不包括任何的反例。

该学习方法有两个假设，其一是示教例子必须是系统所学的概念的例子，要么是正例，要么是反例。如果违反了这一假设，那么就变成了学习多个概念的问题。其二是所要学习的概念一定要能表示成规则空间中的一个点。如果违反了这一假设，那么就必须解决新项问题。

单个概念的学习问题比较简单，并且也比较容易推广到复杂的学习问题中。下面就单个概念的几种学习方法加以介绍。

#### (一) 变型空间法

变型空间法是 T·M·Mitchell 于1977年提出的一种数据驱动型的学习方法。

该方法以整个规则空间为初始的假设规则集合  $H$ 。依据示教例子中的信息，系统对集合  $H$  进行一般化或特殊化处理，逐步缩小集合  $H$ 。最后使得  $H$  收敛到只含有要求的规则。由于被搜索的空间  $H$  逐渐缩小，故称为变型空间法。

##### 1. 规则空间的结构

在规则空间中，表示规则的点与点之间存在着一种由一般到特殊的偏序关系。图8-4表示了一个规则空间偏序关系的一部分。

在图8-4中，TRUE 表示没有任何条件，这是最一般的概念。概念  $\exists x: \text{CLUBS}(x)$  表示至少有一张梅花牌，它比 TRUE 要特殊些。而概念  $\exists x, y: \text{CLUBS}(x) \wedge \text{HEARTS}(y)$  表示至少有一张梅花牌且至少有一张红心牌，它比前两个概念更特殊。图8-4中的箭头用来指向从特殊概念到一般概念。规则空间中最一般的点是没有描述的点，这种点对例子不加任何限制，因而可以

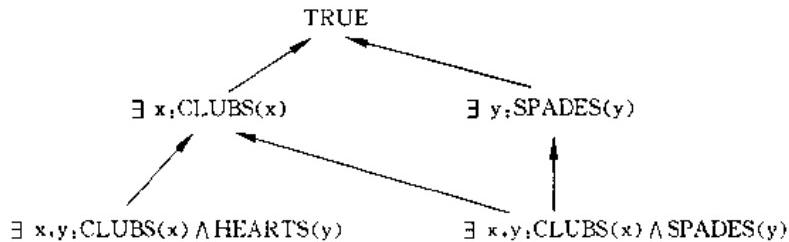
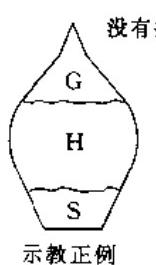


图 8-4 规则空间偏序关系例

描述任何事物，所有实例都符合这一概念。图 8-4 中最下面一行的各点是正例直接对应的概念，每一个点的概念只符合一个正例。现在一手牌，例子是每张牌  $c$  的花色和点数，比如：

$$\text{SUIT}(c, \text{CLUBS}) \wedge \text{RANK}(c, 7)$$

这是一个正例，是一个最特殊的概念。而概念  $\text{RANK}(c, 7)$  是规则空间中部的点，它比没有描述要特殊，比示教正例要一般。



在搜索规则空间时，使用一个可能合理的假设规则的集合  $H$ ，

$H$  是规则空间的子集，从图 8-5 可知， $H$  中最一般的元素构成的子集为  $G$ ，

$H$  中最特殊的元素构成的子集为  $S$ 。在规则空间中， $H$  是以  $G$  为上确界和以  $S$  为下确界的一段。因此，可以用  $G$  和  $S$  来表示集合  $H$ 。

## 2. 修选删除算法

图 8-5 规则空间结构示意图

Mitchell 的学习算法称为候选删除算法。在这种算法中，把尚未被数据排除的假设称为可能假设，把所有可能假设构成的集合  $H$  称为变型空间。

算法一开始，变型空间  $H$  包含所有的概念。随着向程序提供示教正例后，程序就从变型空间中删除候选概念。当变型空间仅包含有一个候选概念时，就找到了所要求的概念。该算法分为四个步骤：

(1) 把  $H$  初始化为整个规则空间。这时  $G$  仅包含空描述。 $S$  包含所有最特殊的概念。实际上，为避免  $S$  集合过大，算法把  $S$  初始化为仅包含第一个示教正例。

(2) 接受一个新的示教例子。如果这个例子是正例，则从  $G$  中删除不包含新例的概念，然后修改  $S$  为由新正例和  $S$  原有元素共同归纳出的最特殊的结果。这个过程称为对集合  $S$  的修改过程。如果这个例子是反例，则从  $S$  中删去包含新例的概念，再对  $G$  作尽量小的特殊化，使之不包含新例。这个过程称为集合  $G$  的修改过程。

(3) 重复步骤②，直到  $G=S$ ，且使这两个集合都只含有一个元素为止。

(4) 输出  $H$  中的概念（即输出  $G$  或  $S$ ）。

下面给出一个候选删除法的实例。在这个例子中，用特征向量描述物体，每个物体有两个特征，即大小和形状。物体的大小用大(lg)或小(sm)来描述。物体的形状用圆(cir)、方(squ)或三角(tri)来描述。若用  $x$  表示大小， $y$  表示形状，那么要教给程序“圆”的概念，就可以表示为  $(x, \text{cir})$ 。按候选删除法的算法步骤是：

步骤①：把集合  $G$  初始化为最一般的概念，把集合  $S$  初始化为仅包含第一个示教正例，即

$$G = \{(x, y)\}$$

$$S = \{(sm, \text{squ}), (sm, \text{cir}), (sm, \text{tri}), (lg, \text{squ}), (lg, \text{cir}), (lg, \text{tri})\}$$

现在提供第一个示教正例  $(sm, cir)$ , 表示小圆是圆。往下进入第②步。

② 经过步骤①后, 得到

$$G = \{(x, y)\}$$

$$S = \{(sm, cir)\}$$

再接受新的示教例子, 这个例子是  $(lg, tri)$ 。这表示大三角不是圆。这是个反例。算法使得  $G$  集合特殊化, 得到:

$$G = \{(x, cir), (sm, y)\}$$

$$S = \{(sm, cir)\}$$

图8-8表示了第二个示教例子加入后的变型空间。这时  $H$  仅含有三个概念。它们是满足第一个例子, 但不满足第二个例子的全部概念。

再执行步骤②, 接受第三个示教例子  $(lg, cir)$ , 这是一个正例, 表示大圆是圆。这一步首先从  $G$  中删除不满足此正例的概念  $(sm, y)$ , 再对  $S$  和该正例作一般化, 得到

$$G = \{(x, cir)\}$$

$$S = \{(x, cir)\}$$

步骤③: 由于此时  $G = S$ , 且两个集合仅包含一个元素, 因此算法结束, 并输出概念  $(x, cir)$ 。

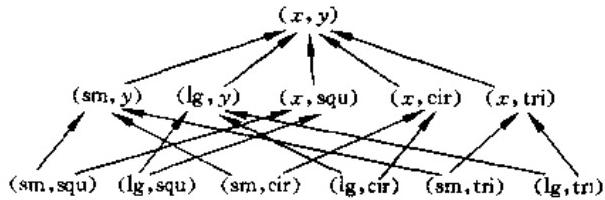


图8-6 初始变型空间

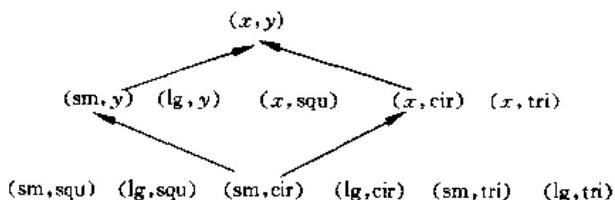


图8-7 第一个示教例子后的变型空间

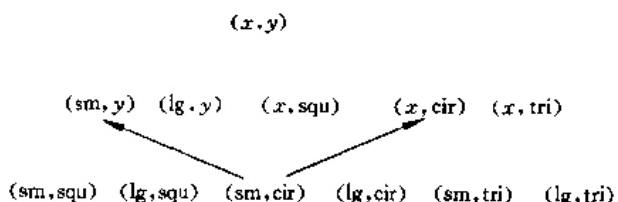


图8-8 第二个示教例子后的变型空间

此算法执行步骤图示分别在图8-6, 图8-7和图8-8中。

### 3. 方法讨论

变型空间法还存在有一些弱点, 需要加以改进。该方法的主要缺点是:

- (1) 抗干扰能力差

变型空间法无法处理有干扰的示教例子，错误的示教例子会对学习过程产生极大的影响，系统可能根本找不到与所有示教例子相容的概念，这时  $H$  成为空集。

对于这种情况，Mitchell(1978年)提出了解决的办法，就是保存多个  $G$  和  $S$  集合。例如， $S_0$  符合所有的正例， $S_1$  符合除一个正例外其他的正例， $S_2, S_3 \dots$  类似。如果  $G_0$  超过  $S_0$ ，则  $H_0$  为空集，这就说明没有任何概念符合全部例子。于是程序再去找  $G_1$  和  $S_1$ ，以便得到  $H_1$ 。如果  $H_1$  也为空集，则再去找  $G_2$  和  $S_2 \dots$  如此继续下去。

## (2) 无法发现析取概念

学习系统中，有些概念是析取的。例如，PARENT 这个概念可能是父亲，也可能是母亲。这表示为：

$$\text{PARENT}(x) = \text{FATHER}(x) \vee \text{PARENT}(x) = \text{MOTHER}(x)$$

由于  $G$  集合和  $S$  集合的元素都是合取形式，所以上述算法找不到析取概念。

对于这个问题，Mitchell 也提出过一种解决办法，即反复使用候选删除算法发现几个合取描述概念。这些合取描述概念同某些示教正例和所有示教反例相容，然后再把这些合取概念析取起来，形成整个概念。

## (二) 精练算子法

精练算子法是一种数据驱动方法，又称为规则空间操作法。该法用于修改规则空间中的假设。程序根据示教例子，用启发式方法选择适当的算子操作。算法的简要过程是：

第一步：收集一些示教例子。

第二步：分析示教例子，以便选择适用的操作。

第三步：由选用的操作来修改假设集合  $H$ 。

第四步：重复第一、二、三步，直到得到满意的假设。

精练算子法的代表性系统是 P·Langley 提出的 BACON(1980年)。该系统用于发现经典物理学定律，如牛顿万有引力定律、欧姆定律和开普勒定律等。

BACON 的思想是：程序反复检查数据，并用改进操作来产生新的项，直到发现一个项总是常量。这样就可以把概念表示为“项=常量”的形式。

### 1. BACON 应用实例

下面通过开普勒定律的发现，考察 BACON 系统的学习。

BACON 用特征向量来表示示教例子，其特征值可为实数、整数或符号。开普勒行星定律是：设行星绕太阳运动的周期为  $p$ ，行星到太阳的距离为  $d$ ，则有关系：

$$d^3/p^2 = k; \quad \text{其中 } k \text{ 是常数}$$

给系统提供的示教例子如表8-1所示。

表8-1 BACON 系统示教举例表

| 例 子            | 行 星 | 特 征 |   |
|----------------|-----|-----|---|
|                |     | p   | d |
| I <sub>1</sub> | 水星  | 1   | 1 |
| I <sub>2</sub> | 金星  | 8   | 4 |
| I <sub>3</sub> | 地球  | 27  | 9 |

BACON 在分析这些示教例子后发现， $p$  和  $d$  都是递增的，且不是线性地依赖，于是它的  
• 170 •

操作就产生新项  $d/p$ , 然后求出  $I_1, I_2$  和  $I_3$  的  $d/p$  值分别为 1.0、0.5 和 0.33。这时, BACON 又发现  $d/p$  是递增的, 于是它又用乘法操作产生新项  $d \cdot (d/p) = d^2/p$ 。此后, BACON 再用乘法得到新项  $(d/p) \cdot (d^2/p) = d^3/p^2$ 。这时程序发现这个项保持为常量, 算法结束, 得到的概念是  $d^3/p^2 = 1$ 。BACON 操作产生的三个新项的值见表 8-2。

表 8-2 BACON 操作产生的新项

| 例 子            | 行 星 | 特 征 |   |      |                   |                                |
|----------------|-----|-----|---|------|-------------------|--------------------------------|
|                |     | p   | d | d/p  | d <sup>2</sup> /p | d <sup>3</sup> /p <sup>2</sup> |
| I <sub>1</sub> | 水星  | 1   | 1 | 1.0  | 1.0               | 1                              |
| I <sub>2</sub> | 金星  | 8   | 4 | 0.5  | 2.0               | 1                              |
| I <sub>3</sub> | 地球  | 27  | 9 | 0.33 | 3.0               | 1                              |

## 2. BACON 的规则空间算子

BACON 的规则空间算子以产生式规则形式存储, 规则左边是测试数据中的模式, 右边是产生的新项。不同的 BACON 程序有着不同的规则空间算子。BACON 的几种算子如下:

- (1) 发现常量 如果某个变量至少两次取相同值, 则建立这个变量等于常量的假设。
- (2) 特殊化 如果建立的假设与数据发生矛盾, 则通过增加合取条件把假设特殊化。
- (3) 产生斜率和截距 如果两个变量保持线性关系, 则产生斜率和截距作为新项。
- (4) 产生乘积 如果两个变量反向变化, 而且斜率不同, 则产生两个变量的乘积作为新项。
- (5) 产生商 如果两个变量同向变化, 但斜率不同, 则产生两个变量的商作为新项。

上述的操作算子是 BACON·1 系统采用的。BACON·2 系统在 BACON·1 的基础上增加了两种操作, 其一用于计算差别以便发现递归序列; 其二用于产生多项式项。BACON·3 是 BACON·1 的另一个扩展系统, 它可以通过发现常量提出假设去重新构造示教例子。

## 3. BACON 系统的优缺点

BACON 系统的优点是能够发现一些用实数值描述变量之间关系的定律和建立新项的功能。

BACON 系统也存在一些问题。首先, 系统对示教例子的取值和变量的次序特别敏感, 以致在有些情况下严重影响甚至完全破坏了学习过程。例如, 用某些示教例子的集合, BACON 就不能发现欧姆定律。其次, BACON 不能处理有干扰的示教例子。例如, 发现常量的算子仅根据某一项在两个示教例子中相等就可以得出结论, 这对干扰是很敏感的。再则 BACON 只能处理涉及数值的学习任务, 而对复杂的逻辑概念无能为力。

## (三) 产生测试法

产生测试法(Generating and Testing)是一种模型驱动方法。代表性的系统是由 T·G·Dietrich 和 R·S·Michalski 研制的 INDUCE1.2(1981年)。该系统的目地是学习积木块世界的结构概念。

### 1. INDUCE1.2 的学习算法

INDUCE1.2 只由示教正例学习单个概念。它试图找出在规则空间中的少数概念, 使得每一个概念都覆盖所有示教例子。程序把示教例子转换为规则空间中的点。这样, 例子和规则就使用统一的表示形式。给系统提供一组示教例子后, 系统从这些实例开始自底向上搜索, 这个过程与候选删除算法中修改 S 集合的过程类似, 通过删除合取条件和增加内部析取, 形成一

般概念的描述。然后再用示教例子对概念描述进行测试，输出那些包括所有示教例子的概念。重复这一过程，直到输出概念达到预定规模为止。算法的具体步骤如下：

第1步：使集合  $H$  初始化为包含示教例子中任一个大小为  $W$  的子集， $W$  称为集束宽度。

第2步：产生更一般化的概念。用一定的方式删掉单个条件，以便使  $H$  中的每个概念都一般化，形成新的集合  $H$ 。

第3步：对集合  $H$  中可能不合理的概念进行修剪，使  $H$  中仅保留  $W$  个概念。修剪的一个准则是根据概念描述的语义特性，如项的数目和用户定义项的费用；另一个准则是使  $H$  中的元素覆盖尽可能多的示教例子。

第4步：对集合  $H$  进行测试。检查  $H$  中每个概念是否覆盖了全部示教例子。如果某个概念是这样，就把它从  $H$  中删除，并放入输出概念的集合  $C$  中。

第5步：重复第2、3、4步，直到  $C$  达到预定规模或者  $H$  变成空为止。

## 2. INDUCE1. 2的应用举例

在积木世界中，积木块的特征可以用一元谓词描述，如分别用  $\text{LARGE}(u)$ 、 $\text{SMALL}(u)$ 、 $\text{CIRCLE}(u)$  和  $\text{SQUARE}(u)$  来描述积木  $u$  是大、小、圆和方等；积木块之间的相互关系可以用二元谓词描述，如分别用  $\text{ONTOP}(u, v)$  和  $\text{SUPPORT}(u, v)$  描述  $u$  在  $v$  的上方和  $u$  被  $v$  支持着等。现在给系统提供两个示教正例。

例1.  $\text{LARGE}(u) \wedge \text{CIRCLE}(u) \wedge \text{LARGE}(v) \wedge \text{CIRCLE}(v) \wedge \text{ONTOP}(u, v)$

例2.  $\text{SMALL}(w) \wedge \text{CIRCLE}(w) \wedge \text{LARGE}(x) \wedge \text{SQUARE}(x) \wedge \text{LARGE}(y) \wedge \text{SQUARE}(y) \wedge \text{ONTOP}(w, x) \wedge \text{ONTOP}(x, y)$

例1表示大圆  $u$  在大圆  $v$  的上方。例2表示小圆  $w$  在大方  $x$  的上方，且  $x$  在大方  $y$  的上方。系统处理示教例子时，采用两空间法。一个空间保存示教例子的特征（一元谓词描述的合取项），一个空间保存示教例子的结构（二元谓词描述的合取项）。系统分别在这两个空间上推广，再组合起来形成所要寻找的概念。本例中，这两个示教正例在结构的规则空间上的投影是：

例1'  $\text{ONTOP}(u, v)$

例2'  $\text{ONTOP}(w, x) \wedge \text{ONTOP}(x, y)$

系统经查找发现， $C = \{\text{ONTOP}(u, v)\}$  是描述结构的唯一概念，它符合所有的示教例子。然后，系统由特征  $u$  和  $v$  推广出新的规则空间。描述这个空间的特征向量是：

$(\text{SIEE}(u), \text{SHAPE}(u), \text{SIEE}(v), \text{SHAPE}(v))$

在这个规则空间中，四个特征分别是  $u$  的大小、 $u$  的形状、 $v$  的大小和  $v$  的形状。把示教例子例1和例2变换到这个空间就得到新的例子。

例1''  $(\text{LARGE}, \text{CIRCLE}, \text{LARGE}, \text{CIRCLE})$

例2. 1''  $(\text{SMALL}, \text{CIRCLE}, \text{LARGE}, \text{SQUARE})$

例2. 2''  $(\text{LARGE}, \text{SQUARE}, \text{LARGE}, \text{SQUARE})$

我们注意到，从例2''得到了两个变换的例子（例2. 1''和2. 2''）。因为  $\text{ONTOP}(u, v)$  可以用两种方式与例2'匹配。一种是  $u$  替换  $w$ ， $v$  替换  $x$ ；另一种是  $u$  替换  $x$ ， $v$  替换  $y$ 。在搜索空间中产生两个覆盖全部例子的概念：

$(\text{LARGE}, *, \text{LARGE})$

和

$(*, \text{CIRCLE}, \text{LARGE})$

其中 \* 表示该特征值可以是任意的。将两个空间的概念组合，就得到完整的概念描述：

$c_1: \text{ONTOP}(u, v) \wedge \text{LARGE}(u) \wedge \text{LARGE}(v)$

$c_1$ : ONTOP( $u, v$ )  $\wedge$  CIRCLE( $u$ )  $\wedge$  LARGE( $v$ )

这两个概念可叙述为：

$c_1$ : 总是一个大物体在另一个大物体的上面。

$c_2$ : 总是一个圆形物体在一个大物体的上面。

### 3. INDUCE1.2系统的优缺点

该系统的优点是比变型空间法快,存储量更少,具有较强的抗干扰能力,只要将要求的概念与全部示教例子相符合改为与大多数示教例子相符合,就可以克服虚假数据的干扰。

INDUCE1.2的缺点是缺乏强有力的模型去指导修剪和结束搜索。再则第二步全部罗列  $H$  中的假设的更一般化,在规模较大的规则空间这样做付出的代价太大。改进方法是仅产生可能合理的假设。另外,由于在算法第三步进行了修剪,所以是不完备的。它不一定能够找到全部合格的概念。第四步又要求同时提供全部例子,因此不适合于逐步学习的情况。

## (四)方案示例法

方案示例法是一种模型驱动方法。它适用于理解性的任务,如图像理解、语音理解和自然语言理解。然而,只有少数学习系统应用了方案示例法。当系统具有大量可以组合起来形成方案,(即抽象的框架规则),学习系统就可以用这种方法。这样,规则空间的搜索可以局限于符合一定方案的子空间。下面介绍一个学习系统 SPARC,它使用方案示例法来发现单个概念。

### 1. SPARC 系统的学习算法

SPARC 学习系统1979年由 Dietterich 研制,用于求解纸牌游戏“Eleusis”中的学习规则。Eleusis 纸牌游戏要求牌手发现由发牌者所确定的秘密规则。秘密规则规定出牌的线性序列,轮到他们时,游戏者试图从他们手中牌打出的牌应按规则延伸这个序列,否则要受罚(再拿一张牌)。发牌者只是指出每人出的牌是否符合规则。最先出完手中牌的人取胜,游戏就结束。图 8-9示出了这种游戏的一个记录。其中每张牌用两个符号表示,第一个符号表示牌的点数,第二个符号表示牌的花色。

|     |    |    |    |    |     |     |    |     |    |    |       |
|-----|----|----|----|----|-----|-----|----|-----|----|----|-------|
| 主行: | 3H | 9S | 4C | 9D | 2C  | 10D | 8H | 7H  | 2C | 5H | ..... |
| 副行: |    | JD |    | AH | AS  |     |    | 10H |    |    |       |
|     |    | 5D |    | 8H | 10S |     |    |     |    |    |       |
|     |    |    |    | QD |     |     |    |     |    |    |       |

规则:若上一张牌是奇数,则出黑牌;

若上一张牌是偶数,则出红牌。

图8-9 Eleusis 游戏的一个记录

在图8-9中,主行记录出对了的牌,副行记录出错了的牌。下面给出秘密规则。两张黑牌是 S(黑桃)和 C(梅花),两张红牌是 H(红桃)和 D(方块)。出牌的次序是 9S, JD, 5D, 4C……,首张牌是 3H,第一张正确牌是 9S,第二张和第三张是错牌 JD 和 5D。

### 2. Eleusis 中的规则

在 Eleusis 游戏中,Dietterich 发现了三种类型的规则,并对每种类型的规则规定了参数化的方案。这三类规则的方案是:

(1)周期性规则 它描述有重复特征的序列。例如,规则“轮流出红牌和黑牌”就是周期性

规则,这类规则的方案可以用  $N$  元组的合取描述为

$$(c_1, c_2, \dots, c_N)$$

其中参数  $N$  是重复周期。上面提到的例子规则可以表示为

$$(\text{RED}(\text{card}_i), \text{BLACK}(\text{card}_i))$$

(2) 分解规则 分解规则用 if-then 规则来描述序列。例如,“如果上一张牌是奇数,则出黑牌;如果上一张牌是偶数,则出红牌”就是分解规则。上面提到的规则可以写成为

$$\text{ODD}(\text{card}_{i-1}) \Rightarrow \text{BLACK}(\text{card}_i)$$

$$\text{EVEN}(\text{card}_{i-1}) \Rightarrow \text{RED}(\text{card}_i)$$

(3) 析取规则 可以用析取范式表示的规则。例如,规则“后一张牌与前一张牌要同点数或同花色”就是析取规则,它可以表示为

$$\text{RANK}(\text{card}_i) = \text{RANK}(\text{card}_{i-1}) \vee$$

$$\text{SUIT}(\text{card}_i) = \text{SUIT}(\text{card}_{i-1})$$

上述每一种方案都有几个参数控制它的使用。除了周期性规则方案中的参数周期  $N$  以外,还有一个参数  $L$ ,称为回顾参数,它指出了要考虑过去的  $L$  张牌。

### 3. 利用方案搜索规则空间

每一个方案都可以看成有它自己的规则空间,SPARC 算法的过程是:

第一步: 方案参数化。SPARC 选择方案并为该方案的参数选定具体值。

第二步: 解释示教例子。把示教例子(即布局中的牌)转换为适应所选择方案的非常特殊的规则。

第三步: 示例方案。把转换了的示教例子通用化以适应该方案。SPARC 使用方案说明算法完成这一步。

第四步: 评价示例的方案。判断方案符合数据的程度如何,删除那些符合例子不良的规则。

SPARC 对所有方案的所有参数空间进行深度优先搜索,直到用户指定的参数数量级限制为止。

例如,当这些步骤应用于图8-8的游戏时,第一步选择  $L=1$  的分解规则方案。第二步把示教例子转换成在规则空间中的具体规则,前五张牌产生下列四个示教例子。每张牌的特征向量表示是

$$(\text{RANK}, \text{SUIT}, \text{COLOR}, \text{PARITY})$$

(SPARC 实际上使用了 24 个特征来描述每一个示教例子,这里仅用了 4 个)

例1: 正例  $(3, \text{hearts}, \text{red odd}) \Rightarrow (9, \text{spades}, \text{black}, \text{odd})$

例2: 反例  $(9, \text{spades}, \text{black}, \text{odd}) \Rightarrow (\text{J}, \text{diamonds}, \text{red}, \text{odd})$

例3: 反例  $(9, \text{spades}, \text{black}, \text{odd}) \Rightarrow (5, \text{diamonds}, \text{red}, \text{odd})$

例4: 正例  $(9, \text{spades}, \text{black}, \text{odd}) \Rightarrow (4, \text{clubs}, \text{black}, \text{even})$

第三步产生下列被示例的方案(其中用 \* 表示无关的特征)。

$$(*, *, *, \text{odd}) \Rightarrow (*, *, \text{black}, *)$$

和

$$(*, *, *, \text{even}) \Rightarrow (*, *, \text{red}, *)$$

第四步判定这两条规则完全符合示教例子,而且句法简单。于是系统就认为这是秘密规则。

### 4. SPARC 的优缺点

方案示例法的优点是可以很快地找到要求的规则。其次是抗干扰性好,可以用统计测量指

导对规则空间的搜索。

它的缺点首先是难以划分方案类型。在 SPARC 中的三种方案虽然覆盖了大多数规则，但也有些规则难以覆盖。其次是对每种方案要建立起相应的算法，这使得它难以应用到新领域。

## 四、学习多个概念

通过学习单个概念的介绍，我们知道，要给系统提供若干正例和反例。程序找到的概念应把例子空间划分为正例区间和反例区间。如图8-10(a)所示。在多个概念学习时，要提供给程序几个概念各自的示教例子。程序由此得到几个概念。每个概念对应例子空间的一个区。如图8-10所示。

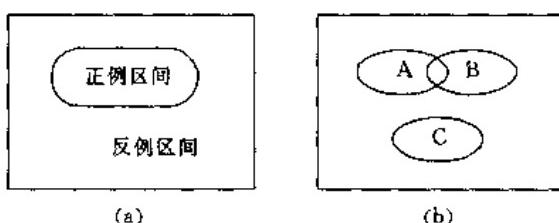


图8-10 例子空间的划分

从示教例子中发现疾病诊断规则的问题是一个重要的多个概念学习问题。例如，诊断多种疾病的程序应具有下列规则集：

(疾病 A 的诊断症状)  $\Rightarrow$  疾病是 A

(疾病 B 的诊断症状)  $\Rightarrow$  疾病是 B

在多个概念学习中遇到的主要问题是重叠概念的描述。例如，在图8-9(b)中，当病人诊状落在 A 和 B 的重叠区域时，应诊断病人有 A 和 B 两种病。在有些情况下，这种重叠是正确的，如疾病诊断往往如此。在另一些情况下，问题要求各个概念要互不相容，如手写体字符的识别就要求以字符作唯一的分类。这时重叠会引起新的问题。为此，在加入新概念时应对原概念进行修改，防止重叠。

下面介绍几个成功的多个概念学习系统。

### (一) AQ11系统

AQ11系统是 R·S·Michalski 等人研制用于学习诊断黄豆病害分类规则的系统。

AQ11寻找规则空间中尽可能一般的规则，使该规则把  $c_i$  类的示教例子与其他的类型  $c_j$  ( $j \neq i$ ) 的示教例子区分开。这些规则称为鉴别规则。

#### 1. 用 AQ 算法寻找鉴别规则

AQ11用 AQ 算法搜索规则空间，它相当于反复应用消除候选算法。它把学习鉴别规则问题转化为一系列学习单个概念问题。为了寻找对  $c_i$  类规则，它把  $c_i$  类的例子作为正例，而把所有其他类的例子作为反例。由此找到覆盖全部正例而不覆盖任一反例的描述，以此作为  $c_i$  的规则。这样找到的鉴别规则集可能在例子空间中未观察的区域内重叠。图8-11表示了这种方法的分类情况。

在图8-11中，圆点表示已知的示教例子，椭圆圈表示鉴别规则划分的区域。

重叠现象的产生使得鉴别规则有二义性，但可以使分类工作做得更稳妥一些。系统的执行

部分可以把这种情况报告给用户,而不是随心所欲地把未知的例子归入某一类。

AQ11用另一种方法寻找不重叠的分类集。为了寻找  $c_i$  类的分类规则,它以  $c_i$  的例子为正例,它的反例包括所有其他类型  $c_j (j \neq i)$  的例子和已处理的各类型  $c_k (1 \leq k \leq i)$  的正例区间中全部正例。于是, $c_2$  类只覆盖  $c_1$  类未覆盖的那一部分, $c_3$  类覆盖的那部分是在  $c_1$  和  $c_2$  类都不覆盖的那一部分中。

图8-12表示了鉴别规则大致对应于符合示教例子的最一般的概念描述,即各类型的  $G$  集合。它还可以求出符合示教例子的最特殊的概念描述,即  $S$  集合。当  $G$  集合和  $S$  集合都可以用时,可以作确定性分类(被  $S$  集合覆盖的例子),可能性分类(仅被一个  $G$  集合覆盖的例子),和多种分类(同时被多个  $G$  集合覆盖的例子)。

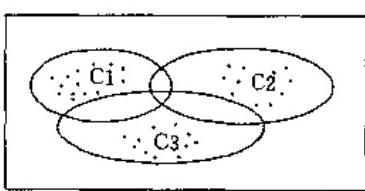


图8-11 寻找允许重叠的鉴别规则

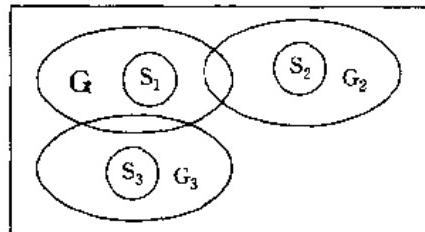


图8-12 各个类型的  $G$  和  $S$  集

## 2. AQ11的应用

AQ11程序已应用于对15种大豆病害的诊断规则。系统提供了有病害的大豆植株的描述(每个描述有35个特征的特征向量)。同时,专家对每一植株的诊断也输入给计算机。系统专门有一个实例选择程序(ESEL)选取了290个样本作为示教例子。ESEL的功能是试图选出和其他的植株样本差别很大的植株样本作为示教例子,而其余的340个植株样本作为测试集合,它们用于检验得到的规则。程序可以由这290个示教例子推出上述可能重叠的规则。

专家所使用的语言比AQ11所使用语言具有更强的表示能力。例如,专家的规则把某些特征列为必要的条件,另一些特征则列为充分条件,但AQ11却不能作这样的划分。尽管如此,由程序产生的规则正确诊断疾病的比率达97.6%,而专家的诊断正确率仅有71.8%。此外,程序驱动规则趋向于给出较少的候补诊断,以避免过多的重选择。

## (二)Meta-DENDRAL

Meta-DENDRAL是发现描述称为质谱仪的化学仪器操作规则的程序。它是DENDRAL专家系统的学习环节,它作为质谱仪仿真程序发现分子分裂规则。化学家注意到,结构形式相同的分子在质谱仪中所表现的行为是相似的。反之,结构形式不相同的分子在质谱仪中所表现的行为是绝然不同的。因此,我们可以按分子结构的类型来对分裂规则进行分类。

Meta-DENDRAL的学习是对于某一类分子结构来发现分裂规则的。该问题可以描述为:

给定: (a) 描述分子结构和子结构的表达语言;

         (b) 选自单一类型分子的示教例子集及其结构和质谱。

寻出: 描述这一类型分子在质谱仪中的分裂规则集。

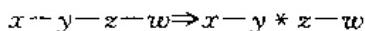
这个学习问题是困难的,因为它包含有两个不确定的因素。其一是示教例子质谱的噪音的干扰。有时可能错误的观察到碎片,造成假的正例,但却对一些重要的碎片而没有观察到,造成假的反例。其二是分裂规则不一定与例子完全一致。只有分裂规则可以预测大多数分子,这

种分裂规则才能被接受。从仿真的观点考虑,DENDRAL认为预测出不发生的分裂比没有预测出要发生的分裂更加安全。

Meta-DENDRAL有两个观点。其一是先作大致的搜索,再作精细的搜索;其二是把学习多个概念问题变换成一系列的学习单个概念问题。

### 1. Meta-DENDRAL 的表示语言

Meta-DENDRAL的表达语言相当于化学家们所用的球和杆模型。分子表示为无向图,其中节点表示原子,而边表示化学键,但它不表示氢原子。每个原子有四个特征,即原子类型(如碳、氮)、非氢邻接点数目、邻接的氢原子数目和双键的数目。分裂规则的条件部分是用上述特征描述的分子结构,分裂规则的执行部分指出将在质谱仪中发生分裂的键。图8-13给出了一个典型的分裂规则。



| 节点类型 | 原子类型 | 邻接节点数 | H 邻接节点 | 双键 |
|------|------|-------|--------|----|
| x    | 碳    | 3     | 1      | 0  |
| y    | 碳    | 2     | 2      | 0  |
| z    | 氮    | 2     | 1      | 0  |
| w    | 碳    | 2     | 2      | 0  |

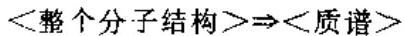
图8-13 典型的分裂规则

仿真程序应用产生式规则,检查是否符合条件部分的分子结构(键环境),若符合则分裂由\*表示的键。

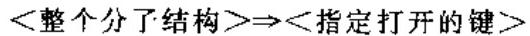
### 2. 解释例子子程序 INTSUM

Meta-DENDRAL采用模型驱动的生成和测试方法搜索可能的分裂规则和规则空间。但是,在它能执行这种搜索之前,它必须首先要用INTSUM子程序解释例子,并把它们转化为规则空间中非常特殊的分裂规则。

示教例子的形式为:



INTSUM则试图提供下面形式非常特殊的分裂规则,形式为:



为了完成这个转换,INTSUM要假设打开哪个键形成质谱中哪个尖峰。在这里,它借助了质谱的半序理论。半序理论把质谱仪的工作描述为分子的一系列分裂。一次分裂把分子分成两部分,以后的分裂再把一部分分成两个小的部分。每部分中的某个原子可能迁移到另一部分中去。半序理论对分裂和迁移的过程给予一定的约束。这些约束是:

- 双键和三键不打断;
- 芳香族环的键不打断;
- 同一个原子的两个键不同时打断;
- 多于三个的键不同时打断;
- 至多发生两次分裂;
- 至多分裂两个环,但需两次分裂。

对迁移或丢失的约束是:

- 至多两个氢原子迁移；
- 至多丢失一个 H<sub>2</sub>O；
- 至多丢失一个 CO。

INTSUM 针对示教例子的分子结构进行仿真的分裂。若得到的仿真谱线符合例子中给出的谱线，则肯定这种分裂，否则就去掉这种分裂。

半序理论对学习问题带来了含糊的因素，解释的一组示教例子可能很容易包含错误例子，这给规则空间的搜索造成了干扰。

### 3. 规则空间的搜索

Meta-DENDRAL 分两个阶段搜索规则空间。首先，RULEGEN 子程序进行模型驱动的生成和测试搜索。这是一种粗略的搜索，它可能得到冗余的或近似的规则。然后，搜索由 RULEMOD 子程序进行。该子程序可以对 RULEGEN 提供的规则进行清理，使它们更为精确并更少冗余。

(1) RULEGEN 子程序 这个子程序按照由一般到特殊的次序搜索规则空间。算法重复地生成一组新的假设规则集  $H$ ，并由 INTSUM 所提供的示教正例对它进行测试。具体步骤是：

步骤1：初始化  $H$ ，使之包含最通用的键环境。

| $x * y$ |      |     |        |    |
|---------|------|-----|--------|----|
| 节点      | 原子类型 | 邻节点 | $H$ 邻点 | 双键 |
| $x$     | 任意   | 任意  | 任意     | 任意 |
| $y$     | 任意   | 任意  | 任意     | 任意 |

这个键环境匹配分子中每一个键，并指出打断每一个键，然后对  $H$  进行特殊化。

步骤2：产生新的假设集。它改变在离开分裂的键（\* 键）一定距离处的原子，由此使  $H$  特殊化。这可以增加新的邻接原子，也可以是确定原子特征。由于修改有一定距离处的所有原子，所以它是粗糙的。

步骤3：用示教例子测试  $H$ 。对每个键环境计算其准则，判断该环境是否比其双亲环境更加合理。在  $H$  中保留比双亲合理的环境，从  $H$  中去掉其他的环境。如果一个双亲环境的所有子环境都不如双亲合理，则输出该双亲的新的分裂规则，并从  $H$  中删除。

步骤4：重复步骤2、3，直到  $H$  为空。

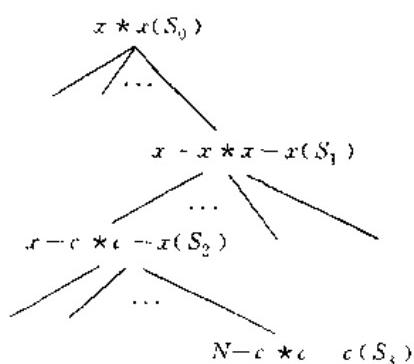


图 8-14 RULEGEN 搜索树的一部分

这种算法假设改进标准单调地增大到单个的最大值（即它是单峰的）。这对于质谱仪的学习往往是正确的。因而，RULEGEN 可以看成是沿单调上升的路径，通过规则空间的偏序直到标准达到局部最大值。

图 8-14 给出了 RULEGEN 搜索树的一部分。

(2) RULEMOD 子程序 RULEGEN 子程序产生的规则仅是近似的，且还未经反例的验证。RULEMOD 子程序使用爬山法在由 RULEGEN 找出的规则空间中，进行搜索，试图改善这些规则。RULEMOD 子程序的执行步骤为：

步骤1：选择重要的规则子集。RULEGEN 可能产生彼此不同但能解释多个数据的规则，RULEMOD 试图寻找一个能满足所有数据的较小的规则

集。它用质谱仪仿真程序测试每个规则，规则按评价函数的值排序，评价函数为：

$$I \times (P+U-2N)$$

其中  $I$  是正确推断峰的数目平均密度， $P$  是正确推断峰的数目， $U$  是仅由这一规则正确推断峰的数目，而  $N$  是不正确推断峰的数目。选择排在最前面的规则，除去由该规则所解释的峰。重复上述的排列和选择过程，直到所有正例被解释，或评价函数的值低于指定的阀值。

步骤2：规则的特殊化，排除反例，RULEMOD 为 EULEGEND 对每个规则未说明的特征值，先检查规则预测的全部正例，得到一张可能的特征值的表，使得这些值适用于所有的正例。子程序 RULEMOD 采用爬山法进行搜索选择可以排除反例的特征值。把那些不相容的特征值从表中去掉。这个过程一直进行下去，直到所有反例都被排除，或不相容的特征值无法再改进。

步骤3：规则一般化以包括正面例证。规则一般化的目的是放松对特征值的要求。RULEMOD 检查规则的键环境中的每个原子，先检查离 \* 键最远的原子，看是否可以去掉整个原子而不引起反面的例证。如果不能去掉，就尝试去掉原子的某个特征。整个搜索过程继续到能做出所有可能的变化为止。

步骤4：选出最后的规则子集。重新应用步骤1所用的过程，以选出最后的规则子集。

#### 4. Meta-DENDRAL 的优缺点

Meta-DENDRAL 是一个有实用意义的学习系统。它对五种类型分子结构发现了分解规则。系统地解决了解释例子问题和在干扰情况下的学习问题。它使用了很多论域知识，即程序中的半序理论。

规则空间的两阶搜索提供了对大型空间搜索的有效方法，同时也提出了多重概念学习怎样转换成一组单个概念学习的途径。

它的缺点是表达方法限于特定的论域以及论域的知识隐含在程序中，而不是表示成显示的知识库。

## 五、学习执行多步任务

前面所讨论的学习问题大多数是如何执行单步任务而进行的，事实上，一种很重要的学习任务是如何完成多步任务。完成多步任务就是选择一个规则序列去完成任务。比如，下棋和符号积分就是多步任务的例子。

我们知道，学习系统的功能环节一是形成学习规则集合，二是确定在什么条件下使用什么规则，即求解策略问题。在这里，着重讨论学习求解的策略问题。

#### 1. 学习多步任务的难点

学习多步任务有两个难点。其一是规则必须联在一起使用，所以必须考虑规则之间的相互影响。例如，在 LEX 系统中，学习环节可能决定：在某些形式为  $\int cf(x)dx$  的积分中，常数  $c$  往往作为因子提到积分号外。这在 LEX 中表达为产生式规则：

OP03：如果有积分形式  $\int cf(x)dx$

那么可以转换为  $c \int f(x)dx$

可是当  $c=0$  或  $c=1$  时，这是一个不可取的步骤，这时应该用规则 OP08 或 OP15。

OP08：如果有被积函数形式 $1 \cdot f(x)$

那么可以转换为 $f(x)$

OP15：如果有被积函数形式 $0 \cdot f(x)$

那么可以转换为0

所以在学习OP03时，应考虑OP08和OP15的相互影响。这时应把OP03改写为：

OP03：如果有积分形式 $\int cf(x)dx, c \neq 1, c \neq 0$

那么可以转换为 $c \int f(x)dx$

第二个难点是奖惩分配问题。在单步任务中，系统应用了执行标准。这个标准可以在系统应用了某条规则之后，立即判断该规则是否正确，从而可以肯定或否定规则。比如在医疗诊断系统中，学习环节可以用专家的确诊作为执行标准，用它来衡量规则的正确性。有时把执行标准直接放入学习过程中，就像在变型空间方法中那样，根据正确的分类来更新变型空间。

在多步任务学习问题中，因为执行标准只能直接确定整个规则序列的正确性。事实上，我们往往更关心序列中每个规则的正确性。由规则序列的正确性确定每个规则的正确性就是奖惩分配问题。例如，一盘棋输了就该否定这盘棋的全局，但这往往因为只是一步或几步走错了而造成的失败，而绝大部分走步还是正确的。我们需要找出是哪几步走错了而造成失败的。

## 2. 执行环节的透明度

执行环节的透明度对于提供学习环节的执行以及考虑所有行动的踪迹是很重要的。一方面它有助于学习环节的奖惩分配。学习环节知道了所选用的每条规则的过程后，就可以确定各个规则的正确性如何。另一方面它有助于把新规则加入知识库中去。学习环节知道了选用规则的细节后，就可以把新规则放入规则集中的适当位置。

## 3. 由执行踪迹抽取局部示教例子

如果仅提供给学习环节一个全局示教例子，它还不能直接由这个例子进行学习。要想学习环节执行这个任务，并比较它的执行结果和例子给出的执行标准，然后由此进行奖惩分配，确定哪些步子是好的，哪些是不好的，这就需要若干局部示教例子。

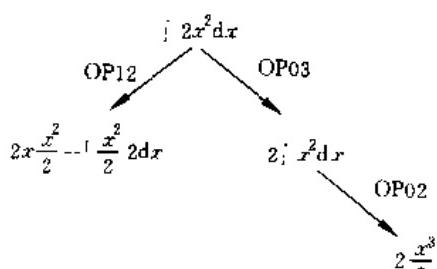


图8-15 一个实例的执行踪迹

例如，在符号积分系统中提供的一个全局示教例子是积分式 $\int 2x^2 dx$  可解，且解为 $\frac{2}{3}x^3$ 。它的执行踪迹如图8-15所示。对执行踪迹进行分析可以发现它的三个局部示教例子。

|                 |        |      |
|-----------------|--------|------|
| $\int 2x^2 dx$  | 用 OP12 | (反例) |
| $\int 2x^2 dx$  | 用 OP03 | (正例) |
| $2 \int x^2 dx$ | 用 OP02 | (正例) |

## 第六节 基于解释的学习

基于解释的学习是利用单个的问题求解例子，由领域知识构造出求解过程的因果解释结构，并获取控制知识，用于指导以后求解类似的问题。

## 一、基于解释学习系统的结构

基于解释学习(Explanation-Based Learning,简称 EBL),也是属于通过实例学习的方法。与通过实例学习的方法的不同之处在于学习系统除了实例之外,还需要具备有关领域的知识,并且能够根据这些知识对实例进行分析,从而构成解释,产生规则。

在 EBL 中,系统的结构由目标概念(target concept)、示教例子(training example)和领域理论(domain theory)组成。目标概念是概念的高级描述,示教例于是系统在学习的过程中提供的一组实例,而领域理论是由一组公理和规则所组成。

EBL 的过程是使用领域理论,解释示教例子是目标概念的一个实例。这个解释构成了示教例子满足目标概念的一个证明。然后,对这个证明进行推广,从而得到目标概念的一个描述。这个描述实际上是满足一个操作性准则。所以基于解释学习的基本结构可以表示为:

给定: 领域理论

    目标概念

    示教例子

    操作性准则

找出: 满足操作性准则的关于目标概念的充分条件。

## 二、基于解释学习的工作原理

1986年 Mitchell、Keller 和 Kedarcabelli 提出了基于解释学习的系统工作步骤,即

1. 产生解释 用户向系统输入实例后,系统首先进行问题求解。比如,由目标引导进行反向推理,从领域知识库中寻找有关规则,使其后件与目标匹配。找到这样的规则后,就把目标作为后件,该规则作为前件,并记录这一因果关系。然后以规则的前件作为子目标,进一步反复分解。如此反复沿着因果链进行,直到求解结束。一旦得到解,便证明了该例的目标是可满足的,由此也得到了证明的因果解释结构。

2. 对解释结构的概括 对所得到的解释结构以及事件进行概括,是采用将常量转换为变量,去掉一些不重要的信息,仅保留求解所必需的那些关键信息,再由组合形成产生式规则,从而获得概括性的控制信息。

## 三、基于解释学习的方法

基于解释的概括(Explanation-Based Generalization,简称 EBG)是 Mitchell 等人综合了以前各种基于解释的概括方法,于1986年提出的一个一般化,独立于具体领域的基于解释的学习方法。

(1) 问题的逻辑描述

系统提供正向、逆向、归纳等多种推理方法。作为一个例子,考虑一个物体能够平稳地放在另一个物体上的问题。示教例子是一个盒子和一个茶几。逻辑描述问题的目标概念是:

SAFE-TO-STACK( $v_1, v_2$ )

其领域知识可表示为:

ON(obj1,obj2)  
 ISA(obj2,ENDtable)  
 COLOR(obj1,red)  
 COLOR(obj2,blue)  
 VOLUME(obj1,1)  
 DENSITY(obj1,1)

领域规则可表示为：

NOT(Fragile(y)) $\rightarrow$ SAFE-TO-STACK(x,y)  
 LIGHTER(x,y) $\rightarrow$ SAFE-TO-STACK(x,y)  
 VOLUME(p<sub>1</sub>,v<sub>2</sub>) $\wedge$ DENSITY(p<sub>1</sub>,d<sub>2</sub>) $\wedge$ X(v<sub>1</sub>,d<sub>1</sub>,w<sub>1</sub>) $\rightarrow$ WEIGHT(p<sub>1</sub>,w<sub>1</sub>)  
 ISA(p<sub>1</sub>,ENDtable) $\rightarrow$ WEIGHT(p<sub>1</sub>,5)  
 WEIGHT(p<sub>1</sub>,w<sub>1</sub>) $\wedge$ WEIGHT(p<sub>2</sub>,w<sub>2</sub>) $\wedge$ LESS(w<sub>1</sub>,w<sub>2</sub>) $\rightarrow$ LIGHTER(p<sub>1</sub>,p<sub>2</sub>)

### (2)产生解释结构

系统从目标出发进行逆向推理，来证明该例满足目标概念，证明的过程中，根据知识库中已有的事实和规则，对目标进行分解。当使用某条规则时，同时返回去把该规则应用到已经变量化的目标概念上。于是，在生成该例求解的解释结构的同时，也生成了变量化了概括的解释结构。图8-16给出了这个示教例子的解释结构。

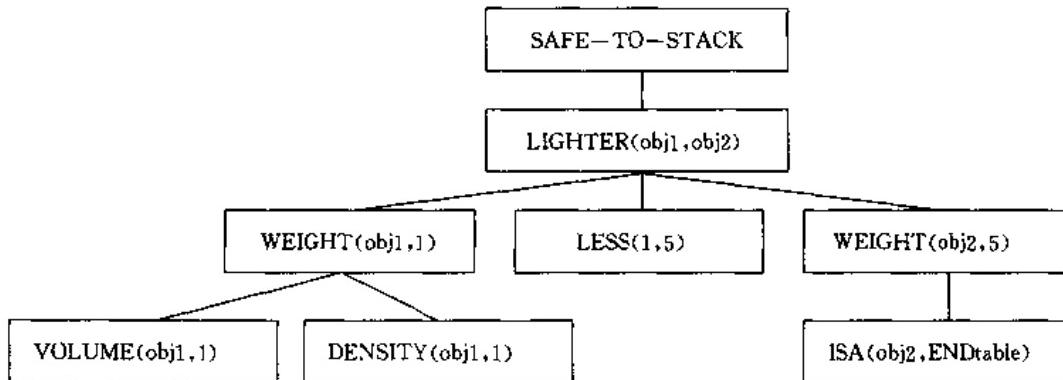


图8-16 示教例子的解释结构

### (3)概括

对产生解释结构步骤中抽出示教例子的有关特性，确立一个充分条件集合。这个条件集合能使得解释结构成立，这个过程就是概括。概括的方法是在解释结构下对目标概念进行回归，回归得到的最后表达式的合取是所求的概念定义。

对上述的问题进行概括，概括的过程如图8-17所示。首先将目标概念 SAFE-TO-STACK(v<sub>1</sub>,v<sub>2</sub>)用规则 LIGHTER(x,y) $\rightarrow$ SAFE-TO-STACK(x,y)回归，以确定 LIGHTER(v<sub>1</sub>,v<sub>2</sub>)是推导 SAFE-TO-STACK(v<sub>1</sub>,v<sub>2</sub>)的一个充分条件，直到解释结构的最后一步。最后，得到回归表达式，即

$VOLUME(v2,v48) \wedge DENSITY(v2,v49) \wedge LESS(v42,5) \wedge ISA(v3,ENDtable) \rightarrow SAFE-TO-STACK(x,y)$

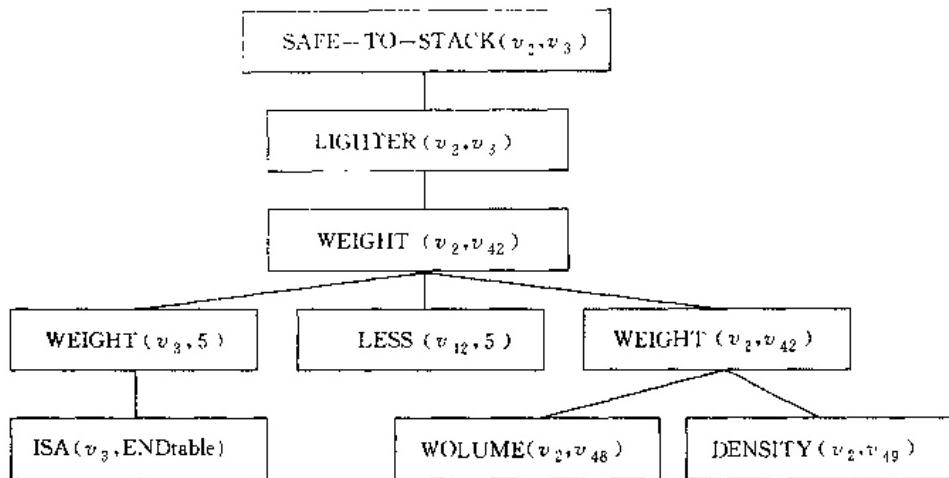


图 8-17 目标概念 SAFE-TO-STACK 的概括过程

## 第七节 从观察中学习

如何对所观察的对象进行有意义的分类是科学上一个普遍的问题,因为这种分类有利于人们对所观察到的事物深刻理解,促进相应科学理论的发展,遗憾的是,尽管人们做了相当的努力,但是对能导致有效分类的机理,到现在还不甚了解。从机器学习的观点看来,形成分类的过程是“从观察中学习”的一种形式。从观察中学习是一种无教师指导的学习,无需示教者对观察的例子分类,因此也不增加人类专家的负担。

产生对象分类的主要思想是对象间相似性的数值化度量。相似性度量通常定义为由对象属性确定的多维空间上的接近度。因此,只有选择出来的属性与对象相似性的描述有关时,这种方法才有意义。然而,数值化度量方法存在有许多缺点。首先这种方法不适应于多值变量和范畴变量,它主要用于数值变量;其次,这种方法只考虑比较对象的特性,没有考虑适用于说明对象特征的上下文内容和概念。为了解决这个问题,产生了概念聚类方法。这种方法是对一些分类对象仅当它们能够被涉及的对象属性的合取概念严格限制时,才能形成一类。下面介绍概念聚类方法。

合取概念聚类问题定义为:

给定:(1)对象的集合

(2)描述对象特征的属性集合

(3)背景知识(包括问题的约束、属性的特征和评价分类质量的标准)

求出:对象的层次分类。每一类由一个合取式描述。任一父辈型的各子类型的描述在逻辑上应不相交。分类体系应使聚类质量标准达到最优。

把对象组织成这样的合取层次结构被称为合取概念聚类。它是一般概念聚类的特殊情形。一般概念聚类定义为描述对象集的概念网络的构造过程,而概念网络把描述对象类的概念作为节点,把类之间的关系作为网络中的链。

Michalski 和 Steep(1981年)提出了概念聚类的思想并确定了合取层次聚类的一般方法,这个算法是在程序 CLUSTER/2中实现的。下面通过 CLUSTER/2的介绍来叙述概念聚类的基本思想。

基本符号和操作如表8-3所示。

表8-3 基本符号和操作

| 符号                                  | 操作                         |
|-------------------------------------|----------------------------|
| $\&$                                | 合取(逻辑与)                    |
| $\vee$                              | 析取(逻辑或)                    |
| $e_i$                               | 事件(对象的描述)                  |
| LEF                                 | 评价函数                       |
| $\text{DOM}(p)$                     | 变量 $p$ 的论域                 |
| $\delta(e_1, e_2)$                  | 事件 $e_1$ 和事件 $e_2$ 之间的句法距离 |
| $\alpha$                            | 复合                         |
| $t$ Complex                         | 逻辑复合                       |
| $S$ -Complex                        | 集合复合                       |
| $E$                                 | 事件空间(事件集合)                 |
| $s(\alpha)$                         | 复合 $\alpha$ 中不可观察事件的数目     |
| $p(\alpha)$                         | 复合 $\alpha$ 中可观察事件的数目      |
| $t(\alpha)$                         | 复合 $\alpha$ 中事件的总数目        |
| $R$                                 | 聚类个数                       |
| $\text{RU}(e_1, \dots, e_n, \dots)$ | 合并操作                       |
| $\text{GEN}(\alpha)$                | 复合 $\alpha$ 的一般化           |
| $\text{COV}(E_1 \setminus E_2)$     | 事件集合 $E_1$ 对 $E_2$ 的覆盖     |
| $G(e \setminus E_0)$                | 事件 $e$ 对事件集合 $E_0$ 的 star  |
| $\text{RG}(e \setminus E_0, m)$     | 约束值为 $m$ 的约束简化 star        |

### 1. 概念内聚

在传统的聚类分析中,两个对象的相似性是靠单一数值描述的。这个数值仅取决于对象的特征,与周围的环境无关。因此,这种相似性度量的方法是与上下文无关的。与此相反,对象  $A$ 、 $B$  间的相似性不仅取决于  $A$ 、 $B$ ,而且还取决于要分类集合的其他对象。例如图8-18中点的聚类问题,人们往往把观察到的点看成是“构成两个菱形”。尽管图中的点  $A$  和  $B$  比其他点相互更加接近些,但它们还是被放在两个不同的类中。这是因为人们是基于概念从属关系进行分类,而不是按距离进行分类。在这个例子中,概念就是“菱形”。

概念聚类将点  $A$ 、 $B$  的相似性度量称为概念内聚,概念内聚不仅取决于  $A$ 、 $B$  两类和附近点集  $E$ ,而且取决于描述  $A$ 、 $B$  整体的概念集  $C$ 。这可以写成:

$$\text{概念内聚}(A, B) = f(A, B, E, C)$$

若假设概念集  $C$  是由一些几何图形组成,如直线、三角形、矩形等,概念内聚的度量可定义为:

$$f(A, B, E, C) = \max_i \left\{ \frac{\# e(i) - 1}{\text{area}(i)} \right\}$$

式中序号  $i$  取遍  $C$  中覆盖点  $A$  和点  $B$  的所有几何图形,  $\# e(i)$  是被图形  $i$  覆盖的  $E$  中所有点的总数,  $\text{area}(i)$  是图形  $i$  的面积。

### 2. 术语和基本操作

### (1) 变量及其类型

设  $x_1, x_2, \dots, x_n$  表示描述对象的离散变量, 每个变量定义一个论域, 且每个论域都是有限集合。因而, 对于变量  $x_i$  的论域可表示为:

$$\text{DOM}(x_i) = \{0, 1, 2, \dots, d_{i-1}\} \quad i=1, 2, \dots, n$$

一般情况下, 论域的大小和结构可以有所不同。变量分为名称变量、线性变量(数值变量)和结构变量三类。它们的论域分别是无序的、全序的和图序的集合。对于结构变量按其一般化的层次结构又分成两类:

①无序的结构变量 层次中叶节点的值形成无序集

②有序的结构变量 层次中叶节点的值形成有序集

图8-19和图8-20分别表示无序和有序的一般化层次结构的例子。

在图8-19中, 叶节点表示特定的形状, 中间节点(多边形、椭圆形、四边形)表示这些形状的一般化。图8-20中, 叶节点表示特定的数量, 中间节点表示这些数量的进一步有序的一般化。

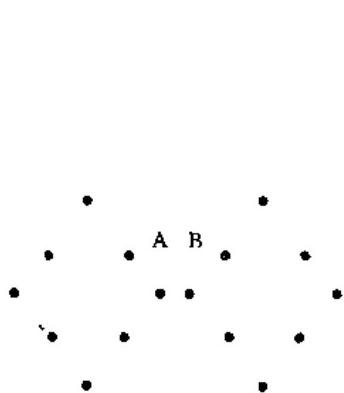


图8-18 概念聚类举例

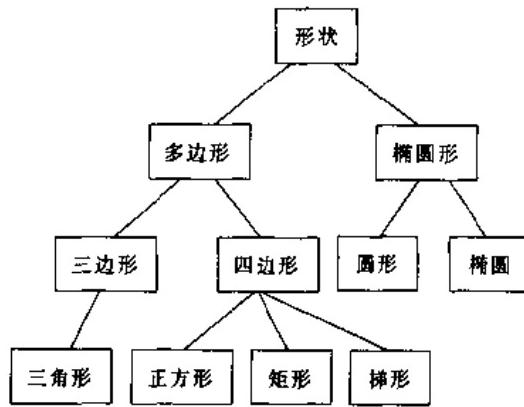


图8-19 一个无序的一般化结构

### (2) 事件和事件空间

事件是对象的具体描述, 其描述形式是变量  $x_1, x_2, \dots, x_n$  的值组成的  $n$  元组。

事件空间是所有可能的事件构成的集合。

### (3) 句法距离

两个事件  $e_1$  和  $e_2$  之间的句法距离  $\delta(e_1, e_2)$  定义为事件  $e_1$  和  $e_2$  中每个变量值之间的句法距离之和。两个变量值之间的句法距离是0到1之间的一个值, 该值取决于变量论域类型的度量方法。对于名称变量, 若值相同则句法距离为0, 否则为1。对于线性变量, 句法距离是两个值差绝对值与该变量论域的总幅度之比。对于结构变量, 句法距离取决于一般化层次结构的类型。无序的一般化层次结构相当于名称变量, 有序的一般化层次结构相当于线性变量。

### (4) 关系命题

关系命题又称为选择器, 其形式为:

$$[x_i \# R_i]$$

其中  $R_i$  是基准。它是变量  $x_i$  论域中元素的表, 表中各元素由析取“∨”进行联结。#是关系符。

它可以是“=”或“ $\neq$ ”, 分别解释为  $x_i$  的值是  $R_i$  中的元素或  $x_i$  不是  $R_i$  中的元素。对于线性变量可以用  $\geq, >, <, \leq$  和区间算符“ $\cdots$ ”。比如, 有下列选择器的例子:

$[length > 2]$   
 $[color = blue \vee red]$   
 $[size \neq medium]$   
 $[weight = 2 \cdots 5]$

### (5) 复合

逻辑复合( $I$ -complex)是选择器的逻辑积。它可以有形式:

$$\&_i \in I[x_i \# R_i] \quad I \subseteq \{1, 2, \dots, n\}$$

如果事件  $e$  中的变量值满足复合中的所有选择器, 那么就说事件  $e$  满足逻辑复合( $I$ -complex)。例如事件

$$e = (2, 7, 0, 1, 5, 4, 6)$$

满足  $I$ -complex  $[x_1 = 2 \vee 3][x_3 \leq 3][x_5 = 3 \cdots 8]$ 。因此,  $I$ -complex 可以看成满足它的所有事件的集合。

集合复合( $s$ -complex)若一组事件存在有一个  $I$ -complex, 该  $I$ -complex 被且仅被这组事件满足时, 则这组事件被称为  $s$ -complex,  $I$ -complex 和  $s$ -complex 统称为复合(complex)。

### (6) 稀疏性

设  $E$  是事件空间, 事件集合  $E \subseteq E^*$  表示被聚类的对象, 则事件  $e \in E$  称为被观察的事件, 事件  $e \notin E$  称为未被观察的事件。假定  $\alpha$  是覆盖一些被观察事件和一些未被观察事件的复合。在  $\alpha$  中被观察事件的数记为  $p(\alpha)$ ,  $\alpha$  中未被观察事件的数目称  $E$  中的绝对稀疏度, 记为  $s(\alpha)$ 。因此, 包含在  $\alpha$  中事件的总数目  $t(\alpha) = p(\alpha) + s(\alpha)$ 。复合的相对稀疏度记为  $r(\alpha)$  被定义为:

$$r(\alpha) = 1 - \frac{p(\alpha)}{t(\alpha)}$$

如果相对稀疏度是 0, 则说明这个复合只覆盖已观察的事件。相对稀疏度是 1, 则说明复合只覆盖未观察的事件。

采用稀疏度作为匹配程度的度量带来的好处是它的简易性, 不足之处是它需考虑整个事件空间。为此引入另一种度量方法, 称为投影稀疏度。它在事件空间的子空间中评价聚类效果, 这个子空间由专门选择的变量来确定。因为不相交聚类中的复合是互不相交的, 所以任意一对复合都至少有一个变量存在有不相交的基准。这种变量称为识别变量, 例如:

$$\begin{aligned} & ([x_1 \geq 3][x_2 = 1 \vee 2][x_3 = 1], \\ & [x_1 < 3][x_3 = 2 \vee 3][x_4 = 3], \\ & [x_2 = 1][x_4 \leq 2]) \end{aligned}$$

它的识别变量是  $x_1, x_3, x_4$ 。

仅遍历识别变量得到的事件空间称为聚类投影事件空间。聚类投影稀疏度等于在投影事件空间中各复合的绝对稀疏度之和。

### (7) 合并算子

合并算子 RU 操作是把几个事件和几个复合合并成一个复合。对于每个变量, 在给定事件的复合中, 变量所取的所有值的集合是确定的。用这些集合作为已产生的复合中的变量基准。例如:

$$e_1 = (2, 3, 0, 1)$$

$$e_2 = (0, 2, 1, 1)$$

和

$$\alpha = [x_1 = 2 \cdot \cdot 3][x_2 = 4][x_3 = 0][x_4 = 2]$$

则它们的合并操作得到的复合

$$\text{RU}(e_1, e_2, \alpha) = \alpha'$$

$$\alpha' = [x_1 = 0 \vee 2 \vee 3][x_2 = 2 \vee 3 \vee 4][x_3 = 0 \vee 1][x_4 = 1 \vee 2]$$

#### (8) 一般化算子

一般化算子 GEN 操作对一个给定的复合进行简化和一般化, 它对复合中每个选择器应用适当的一般化规则。

① 对线性选择器, 使用“封闭区间”规则。这个规则把基准封闭划分为一个或几个不相交的连续区间, 使得未被观察值的数目与区间的长度之比小于或等于一个给定的稀疏阈值。例如若阈值是大于或等于  $3/8$ , 则基准  $1 \vee 2 \vee 3 \vee 7 \vee 8$  可以将区间变成  $1 \cdot \cdot 8$ ; 若阈值是小于  $3/8$ , 则将区间变成两个  $1 \cdot \cdot 3 \vee 7 \cdot \cdot 8$ 。

② 对结构选择器, 可使用“爬山一般化层次结构”规则。在一般化层次结构中, 一个具有多个值的基准可用覆盖基准的一般化层次结构中的大多数具体的节点代替。

③ 在第①步、第②步结束之后, 对各选择或使用“降低条件”规则, 如果在一个选择器中缺省的基准数目与该变量论域中值的数目的比值小于给定的稀疏度阈值, 则去掉该选择器。

为了说明 GEN 算子, 考虑上面给定的复合  $\alpha'$  且假定变量  $x_1, x_2$  是线性的, 变量  $x_3$  是结构化的, 变量  $x_4$  是名称的。 $x_3$  的论域是一般化层次结构, 其中值是值 0, 1 的父节点,  $x_4$  的论域是  $\{0, 1, 2\}$ , 假定对每个变量的阈值都是 0.5, 则我们有

$$\text{GEN}(\alpha) = [x_1 \leq 3][x_2 = 2 \cdot \cdot 4][x_3 = \text{small}]$$

这里, 由封闭区间产生关于  $x_1, x_2$  的基准, 由向上搜索(爬山规则)一般化树产生关于  $x_3$  的基准, 而借助省略条件取消  $x_4$  的选择器。

#### (9) 覆盖

设  $E_1, E_2$  是两个不相交的事件集合, 即  $E_1 \cap E_2 = \emptyset$ 。 $E_1$  对  $E_2$  的覆盖  $\text{COV}(E_1 | E_2)$  是  $S$ -complex 的一个集合  $\{\alpha_j\}_{j \in J}$ , 使得对于每个事件  $e \in E_1$  存在一个  $S$ -complex  $\alpha_j, j \in J$  覆盖该事件, 且没有一个复合  $\alpha_j$  覆盖  $E_2$  中的任何事件, 可形式化为:

$$E_1 \subseteq \bigcup_{j \in J} \alpha_j \subseteq (E^* - E_1)$$

若把覆盖中的各复合  $\alpha_j$  都表示为 l-complex, 则覆盖可以表示为这些复合的析取。

所有  $S$ -complex 是不相交的覆盖称为不相交覆盖。如果  $E_1$  是聚类的事件集合,  $E_2 = \emptyset$ , 则不相交覆盖  $\text{COV}(E_1 | \emptyset)$ , 或简记为  $\text{COV}(E_1)$ , 表示这些事件的不相交聚类。

#### (10) star

下面讨论产生不相交聚类的算法, 这种算法是通过反复构造特殊的覆盖, 这些覆盖称为 star。

设事件  $e$  对子事件集合  $E_0 (e \notin E_0)$  的 star 写成  $G(e | E_0)$  定义为覆盖事件  $e$  且不覆盖  $E_0$  中任何事件的所有最一般复合的集合。所谓“最一般”是指不存在同样的复合  $G'$ , 使得  $G(e | E_0) \subset G'$ 。图 8-21 是 star  $G(e | E_0)$  的例子图示。图 8-21 表示了以线性变量构成的二维空间中, 事件  $e$  对于标记为“•”的事件集的一个 star, 该由 star 复合  $\alpha_1, \alpha_2$  和  $\alpha_3$  组成, star 中不包括  $\alpha'_3$ , 因为  $\alpha'_3 \subset \alpha_3$ 。

下面的算法将对 star 进行两种修改。一种是使星中复合的稀疏度减小。这是由减稀疏度过

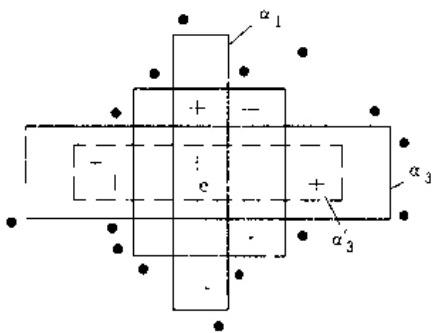


图 8-21  $star G(e|E_0)$  的图示  
说明： $\cdot E_0$  中的元素

+ 在 star 的复合中观察到的事件

为这些复合是覆盖  $e$  且不覆盖  $e_i$  的最大可能的复合。在基本 star 中，复合的数目最大为  $n$ ，由于  $e_i \neq e$ ，所以  $n$  至少为 1。

②确定完全的 star  $G(e|E_0)$ 。假设  $|E_0|=k$ ，首先求出  $k$  个初始的 star 的逻辑积  $\&G'(e|e_i)$ ， $e_i \in E_0$ ，其中  $G'(e|e_i)$  是 star  $G(e|e_i)$  中  $m$  个复合的析取。然后利用吸收律完成复合的合取，直到产生出无冗余的复合的析取。这些复合的集合就是  $G(e|E_0)$ 。

③对  $G(e|E_0)$  中的复合缩小并简化。在 star 中，每个复合的稀疏度在覆盖观察事件的前提下最大可能地减小，这是通过对含有每一复合中的所有被观察事件实行合并实现的。然后，利用 GEN 算子一般化简化各个复合，最后得到的复合正是简化的 star  $RG(e|E_0)$ 。

### (12) NID 过程

NID 过程亦称为不相交化过程。它是把一组相交的复合转换成一组不相交的复合。如果输入到 NID 过程的复合已是不相交的，该过程不起作用。NID 过程的步骤是：

①确定“核心”复合。把多个复合同时所覆盖的已观察事件放入多次覆盖事件表  $m$ -list 中。如果  $m$ -list 表是空的，则复合只是弱相交，即交集中只包含未观察的事件。在这种情况下，该过程结束，得到的复合的组合是弱相交聚类。如果  $m$ -list 非空，则对每个复合由在该复合中，但不在  $m$ -list 中的已观察事件进行合并操作，得到一个合并复合。这样得到的复合称为“核心”复合。

②对  $m$ -list 中的每个事件确定最佳的“宿主”复合。从  $m$ -list 中选出一个事件，用一般化将所有  $r$  个核心复合延拓到足以覆盖该事件的程度，就将该事件加到这些复合中，于是得到  $k$  个修正过的复合。以  $k$  个不同方式，用相应的修正过的复合取代原来初始集合中的一个核心复合，得到一组聚类。然后，根据某一聚类质量标准对这些聚类加以评价，由此确定最佳聚类，去掉其余  $k-1$  种聚类。这个事件所在的核心复合就是它的最佳宿主复合。对  $m$ -list 中的每个事件反复采用上述过程，得到  $k$  个互不相交的复合的集合，它们的并仍然覆盖原来那些相交复合所覆盖的被观察事件。

### 3. 聚类质量的标准

如何衡量聚类的质量是一件困难的事情。尽管如此，人们还是可以给出两个主要的标准。第一个标准是对聚类的形式描述应该简单，使得对对象分类容易，且使类之间的差异变得清晰。第二个标准是分类描述应该符合实际数据。但是要达到精确的符合会导致描述很复杂，因此这两条标准互相矛盾。解决的方法显然是对两者进行权衡。除了这两条标准外，还可以给出其他

程 (redustar) 实现的。另一种是限制这个 star，这是根据上下文有关准则从中选择一定数目的最佳复合。这是由限制 star 过程 (boundstar) 实现的。

### (11) Redustar 过程

在 star  $G(e|E_0)$  中的复合都是最一般的复合。因此，它们能够以超一般的方式描述对象。例如图 8-21 中复合  $a_3$  减小为  $a'_3$ 。Redustar 过程的步骤是：

①确定基本 star  $G(e|e_i)$ ， $e_i \in E_0$ 。首先确定在  $e$  和  $e_i$  中具有不同值的那些变量。不失一般性，我们假设这些变量是  $x_1, x_2, \dots, x_m$ ，且

$$e_i = (r_1, r_2, \dots, r_m, \dots, r_n)$$

则 star  $G(e|e_i)$  中有  $m$  个复合  $[x_j \neq r_j] j=1, 2, \dots, m$ 。因

的准则。例如 CLUSTER/2 就采用了一种组合的方案,它包括下列基本标准:

- (1)聚类和事件的符合
- (2)聚类描述的简单性
- (3)互相聚类差异
- (4)区分度
- (5)维数的降低

聚类与数据之间匹配可用两种方法计算,分别用  $T$  和  $P$  表示。 $T$  是聚类稀疏度的相反数。 $P$  是复合投影稀疏度的相反数。使用相反数的原因是稀疏度越小则匹配的程度就越好。

聚类描述的简单性为描述选择器总数的负值。

衡量类间差异是聚类中每一对复合之间不相交程度之和。任意一对复合的互不相交程度等于在两个复合中去掉相交的选择器后两个复合中的选择器数。例如,一对复合如下:

```
[color = red][size = medium][shape = circle]
[color = blue][size = medium ∨ large]
```

的不相交程度是3,因为5个选择器中有2个相交。这个标准有利于对那些具有多个不同特征的类的划分。它与传统分类法中采取类间最大距离准则相类似。

区分度是在所有聚类中分别辨别的变量数,即在每个聚类描述中有不同值的变量数。

降低维数是以基本维数的负值来度量。基本维数定义为在聚类中区分所有复合所需要的最小的变量数目。

上述标准的定义有一个特点,那就是增加任何一个标准的值都会增加聚类的质量,组合标准采用 LEF 方法(有容差的编辑估算函数)确定。(Michalski, 1980)LEF 定义了一组标准容差对  $(c_1, \tau_1), (c_2, \tau_2), \dots, (c_n, \tau_n)$ , 其中  $c_i$  是从前面所列表中选出的基本标准,  $\tau_i$  是误差阈值 ( $\tau \in [0 \dots 100\%]$ )。第一步用第一个标准  $c_1$  估算所有聚类,保留那些最好的聚类或在阈值  $\tau_1$  所定义的范围内的聚类。下一步用第二个标准  $c_2$  估算余下的聚类,再用  $\tau_2$  进行类似上面的筛选。这个过程一直进行下去,直到得到的聚类只剩下 - 一个,这就是最佳聚类。若过程用完全部  $n$  个标准容差时,则保留的多个聚类质量相当,可从中任选。选取基本标准,它们次序的编排和基本容差的选定均由人来确定。

#### 4. 聚类模块

##### (1)全搜索算法

聚类模块的基本算法流程图如图 8-22 所示。

算法的目标可描述为

给定:  $E$  被聚类的事件集合

$k$  希望的聚类数目

LEF 聚类质量标准

找出: 事件集合的不相交聚类,使得给定的聚类质量标准为最优。

算法的步骤是:

①选取初始的种子。从给定事件集合  $E$  中选出  $k$  个事件作为初始种子,种子的选取可以是随机的或者遵从某个标准。

②由种子建立 star 使用过程 REDUSTAR 为每个种子建立一个简化 star  $RG(e_i | E_0), E_0$  为其余种子的集合。

③由选择并修改 star 中的复合构造一个最优聚类,即  $E$  的不相交覆盖。由  $k$  个 star 中各选

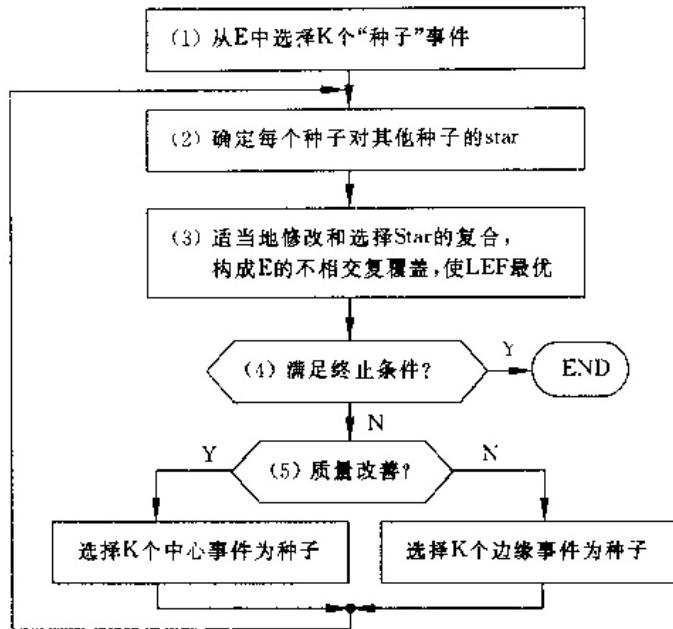


图8-22 聚类模块流程图

取一个复合,这  $k$  个复合构成复合的集合。如果这  $k$  个复合不是互不相交的,则执行不相交化过程 NID,使复合不相交。

④评价终止标准。如果是第一次迭代,存储这个聚类结果。在以后的迭代中,只有根据 LEF 确定比先前存储的聚类质量更好时,才能存储该类结果。在预定数目的迭代还未产生更好的聚类时,算法结束。

⑤选择新的种子。对新产生的聚类中的每个复合(共  $k$  个复合),都有已观察事件集合,从中选择新的种子。选择种子的方法有两种,其一是选择“中心”事件。“中心”事件定义为最靠近复合的中心(由句法距离确定远近);其二是采用“逆行原理”,选择“边缘事件”。边缘事件定义为离中心最近的事件。这有利于选用未作种子的事件,只要聚类在改善,就选择中心事件。如果聚类不改善,就选择边缘事件。

选择完种子后,从第②步开始进行算法的新迭代。算法产生  $k$  个  $l$ -complex 来描述各个聚类,并用 LEF 标准评价聚类。算法的终止条件是一个参数对  $(b, p)$ ,其中  $b$  是算法必要进行的迭代次数,  $p$  是进行  $b$  次后的附加迭代次数。仅当一次迭代产生改善的聚类时,才进行附加迭代。算法中计算量最大的部分是给定  $k$  个种子后构成最佳聚类。

## (2) CLUSTER/2 的路径队列排序搜索

如上述的用种子确定聚类的方法非常简单,不过在解决实际问题时显得效率太低,这主要是因为 star 包含了太多的复合的缘故。当有几个变量和  $k$  个种子时,一个 star 可以有  $n^{k-1}$  个复合。例如  $n=30, k=3$ , 则有  $n^{k-1}=30^2=900$  个复合, 即在搜索树中的任何一节点有 900 条分枝, 有  $900^3=72900$  万片树叶, 虽然吸收律可以消去许多冗余的复合,但是 star 仍然很大。为了解决搜索空间太大的问题,改进的方案是用启发式简化搜索,称之为路径队列排序(Path-Rank-Ordered,简写 PRO)搜索。该方法组合了以下的四种方法。

### ①约束 star(boundstar 过程)

一个 star 中复合的数目限定在固定的数目以下,它保证在搜索树上至多有  $m$  个分枝。一个约束的 star 不是包含从初始 star 产生的随意  $m$  个复合,而是  $m$  个最佳的复合。在产生 star

的每一步,首先简化复合并根据假定的聚类质量标准 LEF 以递减次序排列这些复合。在下一个乘积步骤中,只保留前面  $m$  个复合,这样得到的 star 称为约束 star,并写成  $\text{RG}(e|E_n, m)$ 。采用约束 star 提高了效率,但也失去了最佳聚类的保证,然而这个影响对搜索来说不大,因为每次迭代的聚类仅影响到下一次迭代的种子,所以其最佳性并不重要。

### ② 动态产生 star

由于对复合的估算需依赖于前面已选出的复合,所以在搜索树中,每一个节点上的约束 star 的方法也不相同。CLUSTER/2 采用“偷懒”策略,它是指只有在搜索的路径上需要扩展一个节点时,才产生一个 star。

### ③ 根据路径序号搜索

在约束 star 中的复合是按 LEF 下降次序排列的。在搜索树中,到最佳复合的枝赋给枝序号 0,到下一个最佳复合的枝赋给枝序号 1,依次类推直到枝序号  $m-1$ 。从根节点到叶节点的路径序号等于沿着该路径的分枝序号的总和。由根到叶的一条路径表示一个可能的聚类。对各路径按序号大小次序排列,第一次研究序号为 0 的路径,它只包含每个 star 的最佳复合,其次考虑序号为 1 的路径,这种路径共有  $k$  条。

在搜索中,当路径序号增加时,应考虑搜索终止条件。终止条件有两个参数,即搜索基数  $b$  和搜索试探数  $p$ 。搜索基数  $b$  不断地扩展并计算。此后还要考虑搜索试探的附加路径数  $p$ ,对每条路径都要使用不相交化过程 NID。当新聚类优于以前的聚类时,就保存它并研究另一个搜索试探附加路径。

### ④ 削减搜索树

star 的约束数  $m$  随着路径序号的增加不断减小,因而搜索树更充分地发展为包含“高质量”复合的树。

CLUSTER/2 综合了上述四种方法,它的搜索树如图 8-23 所示。

图 8-23 中, $\alpha_{ij}$  表示 star  $i$  的复合  $j$ ,①,②,⋯ 表示扩展节点的次序,整数 0,1,2,⋯ 表示分枝序号,24,22,⋯ 表示对每个路径的聚类估计得分。

图中  $m$  的最大值为 3,扩展根节点是由 star  $G(\text{种子 } 1 | \text{其他种子}, 3)$  完成的。它的复合是  $\alpha_{11}, \alpha_{12}$  和  $\alpha_{13}$ (它们以 LEF 确定的质量递减的次序排列),表示这些复合的分枝分别赋予分枝序号 0,1,2。下一步要扩展与分枝 0 相连的节点,生成 star  $G(\text{种子 } 2 | \text{其他种子}, 3)$ ,建立复合  $\alpha'_{21}, \alpha'_{22}$  和  $\alpha'_{23}$ 。相应的分枝分别赋予分枝序号 0,1,2。首先考虑路径 0-0,它有最小路径序号 0(图中用粗实线表示)。用 NID 过程处理相应的聚类  $\{\alpha_{11}, \alpha_{21}\}$ ,将最好的结果保留下。然后再考虑路径 0-1,由 NID 过程处理相应的聚类  $\{\alpha_{11}, \alpha_{22}\}$  并加以评价,如果它比原来的聚类好,则保留这个结果。为了试探路径 1-0(具有路径序号 1 的第二条路径),构造 star  $G(\text{种子 } 2 | \text{其他种子}, 2)$ ,它包含复合  $\alpha'_{21}, \alpha'_{22}$ ,估算与当前路径相联系的聚类  $\{\alpha_{12}, \alpha'_{21}\}$ ,假定终止条件是搜索基数  $b=2$  和搜索试探数  $p=2$ ,且得分(已在图中示意出),则考察完第四条路径 0-2 后树的搜索就要终止(因为这个路径没有找到更好的聚类)。路径 0-1 是找到的最好聚类,它的得分是 22。

概念聚类的方法还有分类的动态修改、结构对象的概念聚类等技术,因篇幅限制,此处再不讨论。

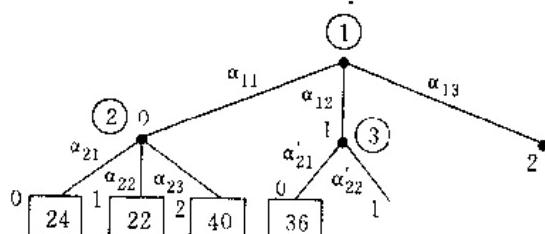


图 8-23 CLUSTER/2 中  $k=2$  的路径序号搜索树

## 习 题 八

1. 什么是学习和机器学习?研究机器学习的意义是什么?
2. 人类学习和机器学习各自的特点是什么?
3. 试叙述机器学习系统的结构和各个组成部分的功能。
4. 试解释机械式学习的模式,机械式学习有哪些重要问题需要加以解决?
5. 试解释学习的基本原理、学习形式和功能。
6. 到目前为止,机器学习方法有哪些?如何对它们进行分类?
7. 叙述机器学习的发展情况。
8. 如果我们对桌子这一概念定义为一切具有大而平的顶部和至少有三条分开腿的物体,试按 Winston 的方法给出学习桌子这一概念的算法。
9. 按变型空间法的思想,给出该方法的 C 语言算法程序。
10. 考虑积木世界的以下属性和属性值:

颜色 = {黄色, 蓝色, 绿色}

形状 = {圆锥形, 球形, 长方形}

硬度 = {硬, 软}

尺寸 = {大, 小}

请采用一种学习单个概念的方法,通过以下实例(用“+”号表示正例,用“-”号表示反例)学习概念:黄色或绿色。实例为:

- (黄色 圆锥形 软 大 +)
- (黄色 球形 软 小 +)
- (黄色 圆锥形 硬 大 +)
- (绿色 球形 硬 大 +)
- (黄色 长方形 软 大 +)
- (蓝色 长方形 软 小 -)
- (蓝色 圆锥形 软 大 -)

11. 试用 Simon 和 Lea 的两空间模型方法,写出学习“彩色电视机”概念的算法。

## 第九章 数据库中的知识发现

### 第一节 引 言

60年代末期以来,随着计算机应用的普及和数据处理在计算机应用中所占比重的上升,数据库技术得到了迅速发展。数据库技术与计算机网络通信已经成为当前计算机应用中两个最重要的基础领域。计算机的一些重要应用,如管理信息系统、办公自动化技术、计算机辅助设计、专家系统等,大都离不开这两个基本技术。

数据库技术是在传统文件技术的基础上发展起来的。数据库技术区别于传统文件技术的特点有:数据共享性、数据独立性、数据操作和控制手段的一致性等。在当前流行的数据库管理系统中,采用的数据模型主要有层次模型、网状模型和关系模型三种。70年代中期以来,关系数据模型逐渐成为占主导地位的数据模型。由于关系数据库的模型结构简单,逻辑物理界面清晰,具有较强的集合处理功能,使得数据库应用系统开发的效率大大提高。

目前,数据库的应用已经触及到人类生活的各个方面,银行、交通、法律、商业、工业、农业、教育、科技、军事、医疗卫生等各行各业都在应用着数据库。据统计,1989年全世界数据库总量为500万个,而且数量以每20个月翻一番的速度增长着,但是对数据库中数据的开发利用还主要是检索查询,效率很低,往往很多数据还没来得及分析就已经过时了。90年代地球探测卫星每天产生的数据,超过以前所有航测数据的总和。即使一个人以最快的速度一刻不停地工作,也要花费几年时间才能浏览卫星一天内所产生的图片。生物学领域研究的数以百万计的遗传基因,世界各国定期进行的人口普查,国土资源地理信息,铁路动态调度控制,公安司法部门的案件处理,都涉及巨量的数据。相当数量的数据具有很强的时效性,数据的价值随着时间的推移而迅速降低。

数据收集与维护的最终目的是为了供人们使用。简单的数据查询或统计虽然可以满足某些低层次的需求,但人们更为需要的是从大量数据资源中挖掘出对各类决策有指导意义的一般知识。它们是对大量数据的高度浓缩和抽象,是对数据整体的全面而深刻的认识。这些经过智能分析和表示的数据才是有价值和竞争力的社会资源。

数据的急剧膨胀和时效性、复杂性远远超过人们的手工处理能力,迫切需要高性能的自动化数据分析工具,高速、全面、深入、有效地加工数据。但目前的情况是,具有智能数据分析的技术仍不成熟,数据产生和处理能力之间的差距越来越大。近年来,各大数据库公司纷纷宣布数据仓库策略,但其着眼点仍然是数据查询,不能从根本上解决数据浓缩问题。一方面,人们苦于不能及时得到科学决策所必须的可靠知识,另一方面,大量宝贵的数据资源甚至还没有得到利用就已经过时。计算机科学当前面临着急需研究和开发发现数据库中知识的技术的挑战。当前的数据库开发利用技术已经不适应现代社会的需求了,需要研究新一代的数据处理技术,以提高信息的利用率。

KDD(Knowledge Discovery in Databases)技术就是在这样一个时代背景下产生的,它的宗

旨就是在数据库中分析处理大量的数据,发现有用的知识,为用户提供所需问题的答案。

## 第二节 KDD 研究现状

KDD一词首次出现在1989年8月举行的第十一届国际联合人工智能学术会议上,常常与“数据挖掘”(Data Mining)混用。KDD就是从数据库中精确抽取大量琐碎数据中隐含的、预先未知的和潜在有用的信息的方法和工具。KDD系统是完成从访问数据库管理系统(DBMS)中的数据、使用模式提取算法、直至评价和解释结果这一知识发现的完整过程的所有组件的集成。KDD与机器学习、统计、智能数据库、知识获取和专家系统一样,都包含领域知识的利用、不确定性管理、交互发现和从研究到应用的转换等方面的研究课题。

由美国人工智能协会(AAAI)主办的首届KDD国际研讨会从1989年起已经召开了四次,会议重点逐渐从发现方法转向系统应用,注重多种发现策略和技术的集成,以及多种学科之间的相互渗透。研讨会主要的议题有:

定性和定量规则的发现(Discovery of Quantitative Laws and Qualitative Laws);

依赖关系与模型的发现与分析(Discovery and Analysis of Dependencies and Models);

文本文件中的知识发现(Discovery in Textual Document);

知识在发现过程中的应用(Using Knowledge in Discovery);

面向具体数据库和领域的发现方法(Database-Specific and Domain-Specific Discovery Methods);

集成与交互式KDD系统(Integrated and Interactive Systems);

所发现知识的应用(Applications of Discovered Knowledge);

另外,数据库、人工智能、信息处理等领域的国际学术研讨会也经常开辟KDD专题。IEEE的Knowledge and Data Engineering会刊也于1993年出版了KDD技术专刊,所发表的5篇论文代表了当前KDD研究的最新成果和动态,较全面地论述了KDD系统的方法论、发现结果的评价、KDD系统设计的逻辑方法,集中讨论了鉴于数据库的动态性冗余、高噪声和不确定性、空值、规则巨大等特点,KDD系统与其他传统的机器学习、专家系统、人工神经网络、统计等系统的联系和区别,以及相应的基本对策。6篇论文摘要展示了KDD在从建立分子模型到设计制造业的具体应用。

根据所发表的KDD论文,现今KDD学术界对KDD理论和技术的基本观点是:

1、一方面要把大量机器学习和相关领域的已有研究成果应用到KDD中,另一方面由于KDD系统处理的数据对象极为复杂,且输出结果必须能够为领域专家理解,所以KDD系统不应该是相关领域成果的简单汇聚。例如:

对机器学习而言,它要求数据库是一种静态的数据聚集,其中的记录通常应该是完整、无噪声的,典型的字段类型为取值种类很少(一般为2)的字符类型。而KDD的数据库通常不符合上述要求。

有的专家系统可以从专家解决问题的实例中归纳出规则,但专家提供的样本质量通常比数据库中的数据质量高得多,样本数据也少得多,因而专家能够确定各种模式的有效性和有用性,而KDD只能依靠DBMS,其自主性受到很大限制。

虽然单纯的统计方法为数据分析提供了坚实的理论基础,但统计方法对许多数据库中取值有限的结构化数据类型是不合适的,大多数统计方法对统计分析的结果常常很难解释。但最

近统计技术正向智能数据分析方向发展,在未来的 KDD 系统中可以起重要作用。

在科学发现中,领域专家可以精心设计实验并获取数据进行分析,数据中含有有关既定分析目标的信息,如果发现最初的实验设计不够合理往往可以重新组织并重作实验。而数据库通常为一般信息获取而设计的,数据库管理者很难为了某个新的研究课题而重复设计数据库并重新采集数据。

基于连接的人工神经网络可以处理复杂的学习问题,而人工神经网络不适合处理结点很多的学习问题,所发现的知识以抽象的[0,1]之间的参数表示,用户很难理解所表示的物理意义。

2、理想的 KDD 系统应该可以自动地处理巨量数据,从中找出最重要、最有意义的分布模式,并把模式转换为适合用户各种领域特定发现专题的知识。但是目前的 KDD 系统距离这种通用、全自动的期望相差甚远。KDD 系统工作的自治性(用机器学习的术语说,就是“无监督学习”)和通用性是一对矛盾,自治要求具备并能够正确使用领域知识,而通用又要求系统与领域无关。解决这个矛盾除了改善发现算法外,更应该从 KDD 中领域知识的表达、获取和使用三个环节上入手。

3、为了提高发现效率,使所发现的知识更切题,应该尽可能多地融入并有效地使用背景知识,充分发挥用户在知识发现过程中的主导作用,尽可能多地利用已经发现了的知识。

4、提高算法效率,包括分布计算、增量计算。KDD 从一诞生起,就带着强烈的应用色彩,吸引了基础研究、应用研究以及信息工业界的普遍重视,希望能够从 KDD 的最新成果中获得经济收益。参加 KDD 研究的人员包括计算机科学家、工程师、认知科学家、数学家、哲学家和许多领域专家。近年来不少差别很大的 KDD 系统相继问世,在医学、金融、农业、社会、市场和销售、保险、军事与公安、空间科学和出版等行业中应用。其中比较有影响的系统有:

DBLEARN 是加拿大 Simon Fraser 大学的韩家卫等人开发的通用知识发现工具。DBLEARN 是基于概念树面向属性进行归纳分析,具有很好的运行效率和一定的抗噪声能力,有较高的自治性,适合分析大型数据库中的数据。缺点是发现模式过于简单。

CoverStory 是一个由美国信息资源公司和 MIT Sloan 管理学院的 J. Little 联合开发的商品化产品,用来从超级市场数据中分析出重要事件,考虑地域、时间等要素,提供有效的解释。到1993年 CoverStory 已经装机几十套。用户输入必要的领域信息后,CoverStory 具有很好的自治性,但其所使用的发现方法应用面较窄。

EXPLORA 是90年代由 P. Hoschka 和 W. Klosgen 开发,能够分析数据,从中发现有用的数据关系。EXPLORA 使用图论理论,通过模式模板网络搜索满足一定统计约束的“事实”,通过很少几类操作来压缩范围,浓缩所发现的事实。用户通过交互式的浏览器裁剪所输出的事实,最终生成发现报告。EXPLORA 适应于经常而又有规律地变动的数据,每次发现前,用户都必须给出分类知识。

Knowledge Discovery Workbench (KDW)是由美国著名的 KDD 专家 G. Piatetsky-Shapiro 领导开发的大型数据库交互发现工具。KDW 通过基于 SQL 的查询界面可以直接访问 DBMS。KDW 知识库所含的信息与数据库有关,用来解释重要的字段组、记录组、函数依赖和 SQL 查询语句。其中的领域知识是自动频繁反复使用的操作脚本。KDW 可以进行聚类、分类、特征描述、偏差检测、强规则依赖关系发现等。KDW 具有良好的领域适应性,但系统自治性较差,需要用户进行较多的定义、控制和评价工作。

CDP 是 IBM Almaden 研究中心的 R. Agrawal 等人开发的原型系统。这个系统主要采用

改进了的 ID3 算法,因而具有较高的执行效率,可以进行多种类型的知识发现。CDP 具有良好的系统自治性,但知识的表现形式相当单调。

DATALOGIC/R 是加拿大 Reduct System 公司开发的专门进行每月股票数据分析的产品。产品所处理数据的噪声很强,所以能够发现与评价弱规则是很重要的。DATALOGIC/R 融入了较多的领域知识,能够对规则作一定的解释。

随着计算机、网络、数据库技术的不断发展和人们对知识和科学决策的迫切需要,KDD 研究和应用会得到很大的发展,并在信息社会中扮演重要的角色。

### 第三节 KDD 的一般机理和理论基础

#### 一、一般机理

推理、联想和学习是人类智能活动的三大主要功能,推理和联想的功能必须通过学习才能不断完善、充实,因而学习是一切智能活动的基础。使计算机系统具有某些程度的学习能力,能够模拟人类的学习活动,一直是人工智能领域所追求的目标。

储存在数据库中的结构化数据,是对现实世界某种程度上的符号化和数据化的抽象,是对现实世界事物某一程度、某一侧面的映射,所使用的抽象方式和抽象层次主要取决于具体的应用模式。考虑到数据采集过程中可能引入误差,因而要求数据库至少能够在总体上反映现实世界,否则,数据库就不能使用了。数据库中的元组可以认为是一些低抽象程度的判断。

那么什么是知识?从数据库中挖掘出的知识具有什么形态?这也是人们常常对 KDD 存有疑问的地方。

知识是通过推理,将已知判断联合起来所产生这一新的判断。例如在一个存有 300 个案犯数据的数据库中,有 A,B,C 三人是学生的事例,进而统计出数据库中有 1% 的人是学生,这并不是知识;只有可以根据这一事实在时间上、空间上外推数据库所代表的另一部分案犯中,也有 1% 左右的人是学生,才能转化为知识。而能否作这种外推,数据库本身不能回答,必须靠用户。

事实上,有很多情况数据库已包容了用户所研究的对象全集,不需要从已知的判断导出数据库外的对象的新判断。但如果能够从数据库中大量的、低抽象度的简单判断,导出用户难以通过观察得到的适当抽象的对象整体特征的模式描述,那么用户就可以通过所发现的模式,结合自己的领域知识,容易得出数据库本身不能给出的新结论,这种模式就是一种知识。

假设在 KDD 系统中就“抢劫案件”主题进行发现,可能给出“抢劫案犯中有相当比例的外来打工人员”的结果。结果也有可能是“抢劫案犯中有相当比例的 20~30 岁人员”,或“集中在东城区的储蓄所附近”,或“被劫对象集中于 30~40 岁的男性”,或“预备手段多为跟随”等,到底存在哪种模式,或哪种模式最突出用户事先并不知道,这种与发现主题有关的数据模式是由系统主动给出。而数据库传统意义上的统计,则是系统被动地对用户提出的数据模式进行某种形式的强度验证。比如给出“外来打工人员抢劫犯”的精确案例数。

显然,如果模式比较复杂,模式搜索空间很大,人工很难高效地做出好的假设。但是,如果系统融入了足够的领域知识和常识,我们就有理由期待系统能够自动地给出数据库间隐藏的模式,并得出某种结论。

特定的数据库通常可以用于很多不同主题的发现，用户个人的发现兴趣也是千差万别，能够灵活地运用大量的背景知识和常识就显得非常困难。系统首先自动地从数据库中提取高质量的特征模式，然后人工地解释这些模式并引出相应的面向问题的结论。这样由于最初模式的启发，使得用户的研究主题不断具体化、深化，导引机器就新的深化主题进一步发现新的特征模式。这种相互启发、相互导引、以用户为主、人机密切配合的发现模式在目前来说是一种比较可行的发现方式。

## 二、主要研究方法

KDD 的主要实施对象是关系数据库，这是因为关系数据库具有归一化的组织结构形式、一体化的查询语言、方便的用户接口、能进行集合处理等优点，而且在各行各业中应用最广泛。另外，关系数据库中各关系之间、各属性之间都是平等的，这种特性便于知识发现过程中的并行运算。由于 KDD 的研究对象比较特殊，一般都是大型的数据库，其中的数据容量往往是一般人工智能系统所不能比拟的，因此，KDD 的研究方法及技术策略就有其鲜明的特色。

首先，在研究方法上，遵循认识的基本过程，即实践—认识—再实践—再认识。KDD 一改过去以演绎逻辑为主的策略，在本质上以归纳逻辑为主，采用从个别到一般，从感性到理性的知识抽象过程。当然，在知识发现过程中，也不能完全抛弃演绎，而是归纳和演绎相结合。

其次，KDD 的技术策略也有其特点。把握事物的规则性是人脑思维的重要功能，精确数学就是这种功能的产物和表现。这种定量的分析和计算在以往的知识发现过程中应用得较多，特别在统计学领域。但是，另一方面，在定量基础上的定性归纳有时也能够深刻地反映问题的本质，并且用较少的代价就能传递足够的信息，对复杂事物作出高效率的判断和推理。所以在知识发现过程中，把定性分析和定量分析相结合也是非常重要的，既采用定量的计算来分析和处理数据，也充分重视定性思维和描述的作用。具体地说，知识发现系统应该用语言值来把握过于复杂无法数值化的量的规则性，通过比较来反映事物在量的规则性上的差异。

## 三、抽取知识的类型和表示

从用户的角度看，从数据库中一般可以进行下面四类模式（即知识）的抽取：

(1) 依赖关系：若其中一项的数据可以预测另一项的数据即  $A \rightarrow B$ ，则称这两项存在依赖关系。当确定的依赖关系不存在时，可以附加不确定性度量： $A \rightarrow (0.95)B$ 。这一类知识可用于数据库中的归一化、查询优化，还可用最小化决策树、搜索数据特例等，甚至可以被系统中其他的发现算法使用。

(2) 分类知识：数据子类的标识知识。子类可由某一现有属性确定，也可由附加的领域知识来定义，KDD 系统基于分类知识的发现任务促进了交互式新型聚类算法的发展，即处理器计算能力、用户知识及可视化工具的有机集成。

(3) 描述性知识：关于类别特征的概括性描述。主要包括两类知识：特征描述知识和区分性知识。特征描述知识是指本类数据所共有的；区分性知识是指本类区别于它类的特性。

(4) 偏差型知识：关于类比差异的描述。包括：标准类中的特例；各类边缘外的孤立点；时序关系上单属性值和集合取值的不同；实际观测值和系统预测值间的显著差别等。

为了让用户更好地理解和使用所获取的知识，知识的表示方法就成为重要的问题。数据特

征模式可以直接提供给用户使用,以得出结论,也可以供系统对新对象进行演绎和类比推理,得出新的结论,即传统意义上的知识。直接提供给用户的特征模式必须以用户可以理解的方式表达。从这个意义上说,从数据库中得到的知识可以是一些产生式规则、函数解析式、自然语言文字报告,也可以是一些图表或其他图形语言的表达形式。供系统自己使用的特征模式必须以机器可以解释执行的方式表达。专家系统的产生式规则、语义网络、决策树及遗传算法的参数等,都可以是知识的表达形式。

## 四、知识发现状态空间

关系数据库中的数据是由若干关系表组织起来的,可以从逻辑上把它们虚拟地连接起来,构成一张单一的二维通用大表(Universal Relation)。通用大表的横向是属性(Attribute),纵向是元组(Tuple)或记录(Record)。通过通用大表接口,外部应用可以将数据库看成是单一的平面文件,这样各种基于属性值的已有机器学习和发现算法都可以直接使用。

根据用户提出的发现算法,利用SQL的映射、投影等功能,可以将与发现任务有关的数据取出来。如果待处理的数据量过大,可以先对数据进行抽样处理。最后形成与发现任务有关的初始数据集。

如果要将所发现的知识直接提供给用户,必须对数据进行浓缩。初始数据集D中的键属性唯一地标识了记录,妨碍了对记录的抽象。如果D中含有单属性的键则删除这个属性。若有复合键,则至少删除一部分属性。采用SQL的汇聚归并完全相同的记录,在记录中增加必要的统计属性,记录D中与各个归并元组完全相同的记录条数(vote),每条归并元组都可能对应D中多条记录,称为宏元组。所有宏元组的集合构成了初始的知识模板。

可以从三个方面对知识模板操作:

属性方向 OA(Attribute Oriented): 对属性之间关系的认识和发现;

宏元组方向 OM(Macro Tuple Oriented): 对各宏元组之间一致性和差异性的认识和发现;

知识模板方向 OT(Template Oriented): 是属性值从微观到宏观的操作,使知识模板上升到抽象级别更高的知识模板。

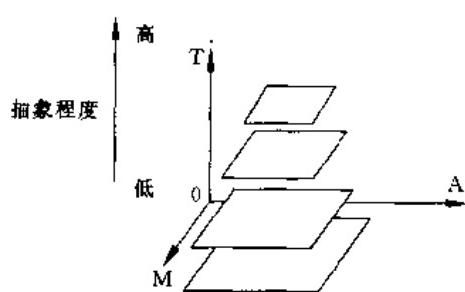


图9-1 知识发现状态空间

对于关系数据库系统的知识发现状态空间是一个三维空间,是发现系统实施多种发现算法的运作空间(如图9-1所示)。

设二维大表由m个属性 $A_i$ 和n个元组组成,每个属性各有 $V_i$ 个不同的取值( $1 \leq V_i \leq n$ ),这样就构成了大表的取值空间:

$$U = V_1 \times V_2 \times \cdots \times V_m$$

每个元组都可以对应到U中的一个实点,n个实点的集合构成R。U中的数据分布模式反映出数据间的作用规律。设 $X_{i,k}$ 为第k个元组的第i个属性的取值,则每个元组都可以用逻辑范式表示:

$$\exists t(t \in R)(A_1(t) = X_{1,k} \cap A_2(t) = X_{2,k} \cap \cdots \cap A_n(t) = X_{n,k})$$

R也可以用逻辑式表达:

$$\begin{aligned}
 \forall t(t \in R) \Leftrightarrow & (A_1(t) = X_{1,1} \cap A_2(t) = X_{2,1} \cap \dots \cap A_m(t) = X_{m,1}) \\
 & \cup (A_1(t) = X_{1,2} \cap A_2(t) = X_{2,2} \cap \dots \cap A_m(t) = X_{m,2}) \\
 & \cup \dots \\
 & \cup (A_1(t) = X_{1,n} \cap A_2(t) = X_{2,n} \cap \dots \cap A_m(t) = X_{m,n})
 \end{aligned}$$

知识发现状态空间中的许多操作都是通过合并、投影等方式,简化关系表的取值描述形式。

## 五、KDD、数据挖掘及其他相关领域之间的关系

从数据库中发现有用模式这一研究课题早在1989年就有人提出,当时人们称作数据库中的知识发现,即 KDD。KDD 这个术语在人工智能和机器学习领域非常盛行。近几年来,随着数据库技术的发展,以及从事该领域研究工作的人员的增加,出现了许多新的与 KDD 相关或相近的术语,如数据挖掘(Data Mining),知识抽取(Knowledge Extraction),信息发现(Information Discovery),信息收集(Information Harvesting),数据模式处理(Data Pattern Processing),数据仓库(Data Warehousing)等。现在,数据挖掘和数据仓库这两个术语被统计人员、数据分析师以及信息管理人员广泛地使用,而且在数据库领域中也越来越流行。这里就 KDD 和它们之间的关系和区别简单地作一个分析。

首先,KDD 指的是从数据库中抽取有用的知识的全过程,而数据挖掘指的是这个过程中的一个特定的步骤,是某个从数据中抽取模式的应用算法。KDD 全过程除了数据挖掘外,还包括数据准备、数据选择、数据清理、调入预备知识、解释发现结果等。盲目的数据挖掘将会导致发现出无意义的模式。其次,KDD 涉及到的研究领域有机器学习、模式识别、数据库、统计、人工智能、专家系统的知识获取、数据可视化,以及高性能的计算等。它总的目标是从大容量的低层次的原数据集中抽取出高层次的知识。这些最终的发现产物应该是可被人所理解的,也是可被再利用的。而数据挖掘则没有这个要求,它的操作结果是一些数据模式,可以用各种能被机器所理解的方式来表示。

另外,与 KDD 密切相关的另一个研究领域是数据仓库。数据仓库指的是为了能够进行在线分析以及获得决策支持而采取的一种收集和贮存数据的新模式。这种新概念在商业领域非常盛行,其中有一种方法称作在线分析处理 OLAP(On-line Analytical Processing)。OLAP 提供多维的数据分析工具,它比 SQL 在计算能力上更强,而且能进行多维计算。OLAP 的目标是为了简化和支持交互式的数据分析,而 KDD 的目标是尽可能地使这个过程自动化。

## 第四节 KDD 系统的基本框架

### 一、KDD 系统的特点

在知识发现过程中,数据库中的数据是供发现的客体,人是发现的主体。人在整个发现过程中处于主导地位,启动、导航并控制整个发现过程。发现系统则是发现工具,是将发现主体和客体相联系的桥梁。由于数据的巨量性、复杂性和易变性,使得用户必须使用高质量的工具才能得到所需要的知识。在发现过程开始时,用户常常并不能提出十分明确的发现目标,只是在

发现的过程中,研究的焦点才会逐渐集中。

作为发现工具的 KDD 系统应该具有下列特点:

- (1)正确理解用户的发现目标要求;
- (2)帮助用户全面了解整个数据库的静态结构和已经具备的领域知识,导引用户选择并补充新的与发现目标有关的领域知识和知识的解释;
- (3)可以发现多种知识,能够满足用户复杂的发现要求;
- (4)有较高的运行效率;
- (5)用户能够随时了解发现过程的进展情况,可以在相当程度上动态控制发现过程;
- (6)能够以用户乐于接受的形式,表达与解释所发现的知识。

## 二、KDD 系统的基本框架

很多人都提出了 KDD 系统的模型,这些模型都具有很类似的结构。一般来说,KDD 系统就是用户和数据库之间的沟通桥梁,系统根据用户要求重新组织、汇聚海量数据,使用户能通过系统对自己感兴趣的模式有清晰的了解,或由系统应用所提取的知识解决具体实际问题。综合目前大多数 KDD 应用产品,我们给出以下 KDD 系统的基本框架(如图 9-2 所示)。

发现控制模块:初始化系统中的其他构件,控制发现活动的启动,状态转移和终止。

数据库接口:响应查询请求,执行 SQL 操作,对原始数据进行加工处理。

背景知识基:与特定领域相关的背景知识库。

汇聚模块:通过对用户任务需求的理解,利用各种数据库操作、数据统计操作,完成初始知识模板的生成。

数据模式抽取模块:集成各种发现算法,对知识发现状态空间执行各种操作,完成知识发现任务。

模式评价模块:依据模式精确度、准确度等评价准则,对发现过程中的中间模式集进行筛选。

知识验证模块:结合各种策略,解决结果知识中的冗余和冲突,达到去伪存真的目的。

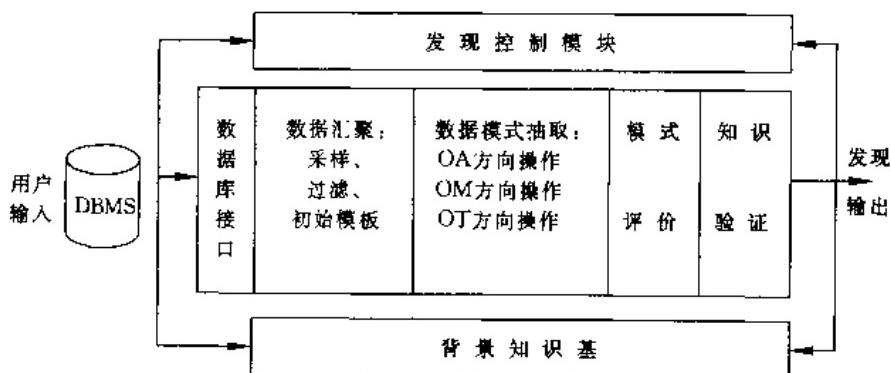


图 9-2 KDD 系统框架图

系统信息来源由两部分组成,一方面用户提出发现任务,并直接干预发现活动,输入各种高级控制指令和各种特定领域背景知识;另一方面数据库管理系统提供原始数据。系统的知识发现过程为:从数据库中选择的数据先经过汇聚模块的过滤形成初始知识模板,再经过发现

控制模块调用各种学习算法形成代表模块集。这些模式被评价模块选择形成最终的知识发现结果。最后，知识验证模块对上述结果进行验证，然后送入系统知识库以支持后续的知识发现活动。

## 第五节 数据库发现知识的方法

知识发现的主要任务在于找出事物之间的因果关系，所采用的核心手段是归纳。归纳法一般分为完全归纳法和不完全归纳法两种。KDD 系统本身并不能自己判断数据库中数据对象是否为同类对象的全体，系统一般把数据库中的数据理解为该类对象的一切特殊情形，并按照完全的归纳法进行操作。

近代归纳逻辑理论集大成者穆勒把发现事物因果联系的方法总结为“五法”：

“求同法”是辨别一类事物的必要条件的方法。将这类事物可能的必要条件列出，找出领先或伴随事物所有特例出现的必要条件。如果存在这种情况，且检查过足够多的实例后，没有发现例外，则可以以一定的置信度相信，这些可能的必要条件就是这类事物的必要条件。

“差异法”是辨别一类事物的充分条件的方法，以一定的置信度找出一类事物的特有的伴随条件。

“同异并用法”是辨别一类事物的充分必要条件的方法。合并使用求同法和差异法可以以一定的置信度找出一类事物充分必要条件。

“剩余法”是差异法的特殊变形，实质上也是以一定的置信度找出一类事物的特有的伴随条件。

“共变法”适用于因果之间存在量的变化的情况。如果一条件发生一定的变化，则事物也必跟随发生某种变化，则这一条件是该事物的原因。以一定的置信度找出这样的条件。

其中后三种方法只是“求同法”和“差异法”的结合、引申和补充，KDD 系统主要用到的也是这两种方法。在实际数据库中，特别是在社会科学数据库中，由于影响对象的因素很多且不易准确地测量对象，同类对象内部的聚合性与异类对象之间的差异性往往并不很强，很难找到“无一例外”的有实用意义的充分条件和必要条件。但是，在很多情况下，具有一定强度的充分或必要条件对于用户还是很有意义的。对于有100个对象的对象类  $B$ ，如果  $A$  伴随99个对象出现，则说  $A$  是  $B$  的很强的必要条件。那一个反例只是削弱了判断的强度，而不是像一般归纳逻辑那样完全否定该判断。因此，KDD 系统主要发现具有一定强度的知识。在有些情况下满足这样的有实用意义的知识也太少，则可以考虑发现具有一定相对强度的知识。条件  $A$  对于对象类  $B$  可能不够强，但对于其他对象类  $A$  还要弱得多。在数据模式规律性很弱的数据库中，这样的知识往往也是很有意义的。

传统归纳逻辑的置信度是指在不完全的归纳中，从部分对象都具有的特性推断对象全体也具有同样的特性。这种推断具有一定的或然性，所观察的特例越多，或然性越低。

KDD 另一个与方法相关的重要研究问题是模糊信息的处理。很多概念，特别是复杂概念的外延不是可以明确限定的，这些概念的界定不完全是由客观的尺度度量的，有相当比例的主观心理因素在起作用。如“年轻”、“冷”和“热”等。事实上，即使是可以采用客观尺度度量的事物属性，由于存在度量精度问题，往往也在一定程度上存在模糊性。模糊数学采用隶属函数描述客观事物在性态和类属方面的这种亦此亦彼性。但是，在隶属函数的实质和具体确定方法这些最基本的问题上，长期以来没有完全解决。

## 第六节 KDD 所面临的问题和研究方向

KDD 是一个新兴的研究领域,它是综合运用人工智能、统计和概率、数据库等成熟技术的交叉学科。它的出现及其在相应的应用系统的开发与应用,为信息社会提供了一个强有力的信息处理工具。

但是,数据库中的真实数据作为 KDD 数据源,存在一些对知识发现不利的情况,主要有以下几点:

- 1.数据库的数据往往是动态的,这使得模式抽取有一定的不稳定性,即有一定的风险;
- 2.数据库中常常有噪声,即存在一些并不反映事实的记录,这将影响抽取模式的准确性;
- 3.数据库中的数据不完全,有些记录的属性域存在空值现象,这使得定焦工作比较困难;
- 4.数据库中信息大量冗余,这样抽取出来的模式对一般用户并不感兴趣;
- 5.数据库中数据稀疏,不能反映事实的整体情况,于是抽取模式如大海捞针;
- 6.现实数据库的数据容量一般非常庞大,这也给知识获取带来很大的麻烦。

在实际数据库中有时只存在一种这样不利的因素,但更多情况下是几种因素都存在,这就要求在设计一个 KDD 应用系统时,要面向数据库的实际情况,有的放矢,针对主要存在的不利因素设计系统结构和算法,使得系统能解决实际问题。

KDD 未来的发展方向可能集中于以下几个方面:

- 1.加速 KDD 系统与各种新技术的融合,如 Internet、数据仓库、主动式数据库、面向对象数据库、特种数据库等;
- 2.进一步研究人类学习的机理,吸纳相关领域的研究成果,解决用户发现任务理解和发现结果呈现两个关键环节中的人机交互问题;
- 3.进一步丰富发现算法,多角度多侧面地发掘数据,特别研究从大量多媒体等非规格化数据中提取知识的有效方法。

随着计算机和数据库等信息处理技术的飞速发展,KDD 必将在信息处理和知识获取中发挥越来越重要的作用。

## 习 题 九

1. KDD 的一般机理是什么? 它的理论基础哪些?
2. KDD 的研究方法有哪些? 各有什么特点?
3. 从数据库中发现的知识主要可以分为几类? 请举例说明。
4. 试述数据库中的知识发现(KDD)、数据挖掘(Data Mining)、数据仓库(Data Warehousing)三者之间的区别和联系。
5. 一个完整的 KDD 系统应该包括哪几大部分? 其主要功能是什么?

# 第十章 遗传算法

## 第一节 遗传算法的基本概念

遗传算法(Genetic Algorithms,简称GA)是人工智能的重要新分支,是基于达尔文进化论,在计算机上模拟生命进化机制而发展起来的一门新学科。它根据适者生存,优胜劣汰等自然进化规则来进行搜索计算和问题求解。对许多用传统数学难以解决或明显失效的复杂问题,特别是优化问题,GA 提供了一个行之有效的新途径,也为人工智能的研究带来了新的生机。GA 由美国 J. H. Holland 博士1975年提出,当时并没有引起学术界的关注,因而发展比较缓慢。从80年代中期开始,随着人工智能的发展和计算机技术的进步,遗传算法逐步成熟,应用日渐增多,不仅应用于人工智能领域(如机器学习和神经网络),也开始在工业系统,如控制、机械、土木、电力工程中得到成功应用,显示出了诱人的前景。与此同时,GA 也得到了国际学术界的普遍肯定。从1985年至今国际上已举行了五届遗传算法和进化计算会议,第一本《进化计算》杂志1993年在 MIT 创刊,1994年 IEEE 神经网络汇刊出版了进化规划理论及应用专集,同年 IEEE 将神经网络,模糊系统,进化计算三个国际会议合并为'94IEEE 全球计算智能大会(WCCI),会上发表进化计算方面的论文255篇,引起了国际学术界的广泛关注。目前,GA 已在组合优化问题求解、自适应控制、程序自动生成、机器学习、神经网络训练、人工生命研究、经济预测等领域取得了令人著目的应用成果,GA 也成为当前人工智能及其应用的热门课题。

## 第二节 简单遗传算法

GA 是基于自然选择,在计算机上模拟生物进化机制的寻优搜索算法。在自然界的演化过程中,生物体通过遗传(传种接代,后代与父辈非常相像)、变异(后代与父辈又不完全相像)来适应外界环境,一代又一代地优胜劣汰,发展进化。GA 则模拟了上述进化现象。它把搜索空间(欲求解问题的解空间)映射为遗传空间,即把每一个可能的解编码为一个向量(二进制或十进制数字串),称为一个染色体(chromosome,或个体),向量的每一个元素称为基因(genes)。所有染色体组成群体(population,或集团)。并按预定的目标函数(或某种评价指标,如商业经营中的利润、工程项目中的最小费用等)对每个染色体进行评价,根据其结果给出一个适应度的值。算法开始时先随机地产生一些染色体(欲求解问题的候选解),计算其适应度,根据适应度对诸染色体进行选择、交换、变异等遗传操作,剔除适应度低(性能不佳)的染色体,留下适应度高(性能优良)的染色体,从而得到新的群体。由于新群体的成员是上一代群体的优秀者,继承了上一代的优良性态,因而明显优于上一代.GA 就这样反复迭代,向着更优解的方向进化,直至满足某种预定的优化指标。上述 GA 的工作过程可用图10-1简要描述。

简单遗传算法的三个基本运算是选择、交换、变异,下而详细介绍。

### 1. 选择运算

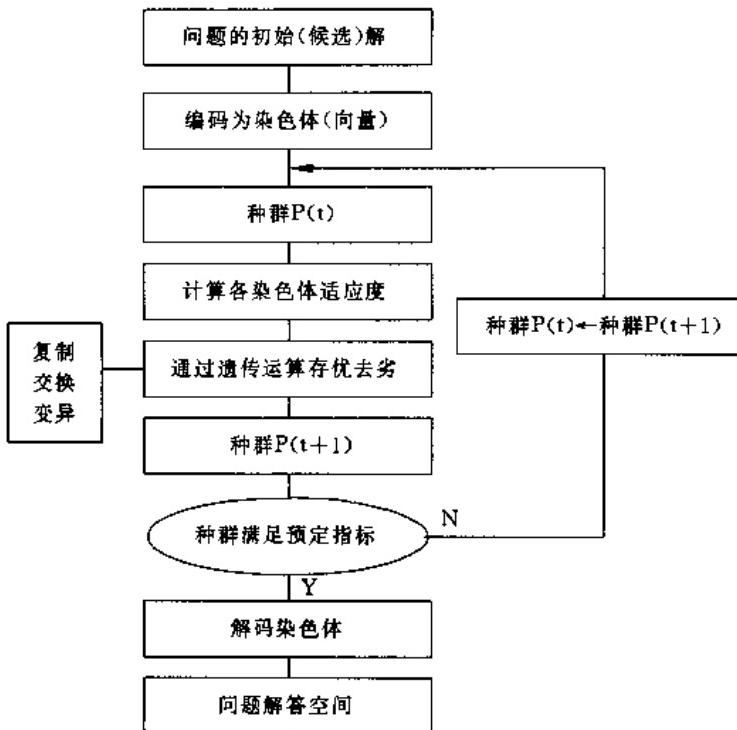


图10-1 遗传算法工作原理示意图

选择运算又称为繁殖、再生,或复制运算,用于模拟生物界去劣存优的自然选择现象。它从旧种群中选择出适应性强的某些染色体,放入匹配集(缓冲区),为染色体交换和变异运算产生新种群作准备。适应度越高的染色体被选择的可能性越大,其遗传基因在下一代群体中的分布就越广,其子孙在下一代出现的数量就越多。有多种选择方法,使用比较普遍的一种是适应度比例法,简述如下。

适应度比例法又称为轮转法,它把种群中所有染色体适应度的总和看作一个轮子的圆周,而每个染色体按其适应度在总和中所占的比例占据轮子的一个扇区。每次染色体的选择可看作轮子的一次随机转动,它转到那个扇区停下来,那个扇区对应的染色体就被选中。尽管这种选择方法是随机的,但它与各染色体适应度成比例。这是因为适应度大的染色体占据轮子扇区面积大,被选中的概率就高(机会多),适应度小的染色体占的扇区小,被选中的概率就低。某一染色体被选中的概率(选择概率)为

$$P_i = f(x_i) / \sum f(x_i)$$

式中  $x_i$  为种群中第  $i$  个染色体对应的数字串,  $f(x_i)$  是第  $i$  个染色体的适应度值,  $\sum f(x_i)$  是种群中所有染色体的适应度值之和。

用适应度比例法进行选择时,首先计算每个染色体的适应度,然后按比例于各染色体适应度的概率选择进入交换(匹配)集的染色体,其具体步骤如下:

(1)计算每个染色体的适应度值  $f(x_i)$ ;

(2)累加所有染色体的适应度值,得最终累加值  $SUM = \sum f(x_i)$ ,记录对应于每个染色体的中间累加值  $S\_mid$ ;

(3)产生一个随机数  $N$ ,  $0 < N < SUM$ ;

(4)选择其对应的中间累加值  $S\_mid \geq N$  的第一个染色体进入交换集。

(5)重复(3),(4),直到交换集中包含足够多的染色体数字串为止。

重复上述过程,直到交换集中包含足够多的染色体为止。显然,此法要求染色体的适应度值应为正值。

例:某种群包含10个染色体,按适应度比例法选择进入交换集的过程示于表10-1及表10-2。

表10-1

| 染色体编号  | 1   | 2    | 3    | 4    | 5    | 6    | 7    | 8    | 9    | 10   |
|--------|-----|------|------|------|------|------|------|------|------|------|
| 适应度    | 8   | 2    | 17   | 7    | 2    | 12   | 11   | 7    | 3    | 7    |
| 选择概率   | 0.1 | 0.02 | 0.22 | 0.09 | 0.02 | 0.16 | 0.14 | 0.09 | 0.03 | 0.09 |
| 适应度累加值 | 8   | 10   | 27   | 34   | 36   | 48   | 59   | 66   | 69   | 76   |

表10-2

|        |    |    |    |    |   |    |    |
|--------|----|----|----|----|---|----|----|
| 随机数    | 23 | 49 | 76 | 13 | 1 | 27 | 57 |
| 所选染色体号 | 3  | 7  | 10 | 3  | 1 | 3  | 7  |

由表10-1,表10-2可以看到,3号染色体的选择概率最高,被选中的次数最多,其他选择概率较低的染色体被选中的次数就少。当然,用适应度比例法进行选择时,性能最坏的染色体也可能被选择,但概率极小,当种群长度较大时,这种情况可以忽略不计。

## 2. 交换

复制操作虽然能够从旧种群中选择出优秀者,但不能创造新的染色体,因此,遗传算法的开创者提出了交换操作。它模拟生物进化过程中的繁殖现象,通过两个染色体的交换组合,来产生新的优良品种,即:在匹配集中任选两个染色体(称为双亲的染色体);随机选择一点或多点交换点位置  $J$  ( $0 < J < L$ ,  $L$  是染色体数字串的长度);交换双亲染色体交换点右边的部分,即可得到两个新的(下一代)染色体数字串。也就是说,交换操作能够创造新的染色体(子孙染色体),从而允许测试在搜索空间中的新点。交换也体现了自然界中信息交换的思想。例如,从匹配集中取出的一对染色体为:

染色体  $A$     11010|110

染色体  $B$     01011|001

随机产生的一点交换位置是5,交换染色体  $A, B$  中第5位右边的部分——110和001,则得两个下一代(子孙)染色体数字串:

$A'$   11010001,

$B'$   01011110。

## 3. 变异

变异运算是用来模拟生物在自然的遗传环境中由于各种偶然因素引起的基因突变,它以很小的概率随机地改变遗传基因(表示染色体的符号串的某一位)的值。在染色体以二进制编码的系统中,它随机地将染色体的某一个基因由1变成0,或由0变成1。若只有选择和交换,而没有变异操作,则无法在初始基因组合以外的空间进行搜索,使进化过程在早期就陷入局部解而中止进化过程,从而使解的质量受到很大限制。通过变异操作,可确保群体中遗传基因类型的多样性,以使搜索能在尽可能大的空间中进行,避免丢失在搜索中有用的遗传信息而陷入局部解,获得质量较高的优化解答。

例1：计算机公司的经营策略优化问题。一个计算机公司追求的目标是最大的利润，为达此目标，必须选择适当的经营策略。一种可能的策略是对以下三个问题作出决策：

- 每台PC计算机的价格定为7000元（低价）还是10000元（高价）；
- 与PC机配套的免费软件是Windows95还是MS DOS；
- 提供快速技术服务（对用户的服务请求立即响应），还是慢速排队服务；

下面用遗传算法来解决这个决策优化问题。

(1) 把问题的可能解表示为染色体数字串。因为有三个决策变量，其取值为是或否，可用1或0来表示，于是，可用三位的二进制数字表征一种可能的经营策略，解的搜索空间为 $2^3=8$ ，即共有8种经营策略可供选择，表10-3示出了其中计算机公司经理已知的4种经营策略（初始解答）。表中数字串的第一位取0表示高价，1表示低价；第二位取0表示配套 Windows95，1表示配套 MS DOS；第三位取0表示排队慢速技术服务，1表示快速技术服务。

表10-3 问题的初始解答

| 序号 | 价格 | 配套软件      | 服务质量 | 染色体数字串 |
|----|----|-----------|------|--------|
| 1  | 高  | MS DOS    | 快    | 011    |
| 2  | 高  | Windows95 | 快    | 001    |
| 3  | 低  | MS DOS    | 慢    | 110    |
| 4  | 高  | Windows95 | 慢    | 010    |

(2)求各染色体的适应度。在这个问题中，一个染色体的适应度恰好是其二进制数等价的十进制数，也就是对应的经营策略的利润（总营业额的百分数）。

表10-4 第0代种群的适应度

| 序号 <i>i</i> | 染色体串 $x_i$ | 适应度 $f(x_i)$ |
|-------------|------------|--------------|
| 1           | 011        | 3            |
| 2           | 001        | 1            |
| 3           | 110        | 6            |
| 4           | 010        | 2            |
| 适应度总和       |            | 12           |
| 最坏适应度       |            | 1            |
| 最好适应度       |            | 6            |
| 平均适应度       |            | 3            |

### (3)选择进入交换集的染色体

交换集又称为匹配池，它实际上是一片内存缓冲区，用来存储欲参加交换的染色体。由表10-4可知，所有染色体串适应度的总和是12，串110的适应度是6，占适应度总和的1/2，也就是串110被选中的机会有两次，其概率为1/2；串011,001,010被选中的概率分别为3/12=1/4,1/12,2/12=1/6。按前述适应度比例法，选择进入交换集的染色体串及其适应度情况如表10-5所示。

由表10-5可知，选择操作的作用是改进了种群的平均适应度，使其由原来的3提高到了4.25，最坏适应度由原来的1改进为2，性能最差的染色体已从种群中删除。但是，选择不能创造新的染色体，为了改进这个缺点，寻找搜索空间内的新的检测点，须进行交换操作。

表10-5 第0代种群的适应度

| 序号 <i>i</i> | 交换集中的染色体串 $x_i$ | 适应度 $f(x_i)$ |
|-------------|-----------------|--------------|
| 1           | 011             | 3            |
| 2           | 110             | 6            |
| 3           | 110             | 6            |
| 4           | 010             | 2            |
| 适应度总和       |                 | 17           |
| 最坏适应度       |                 | 2            |
| 最好适应度       |                 | 6            |
| 平均适应度       |                 | 4.25         |

#### (4) 交换操作

设采用单点交换,随机产生的交换点是2,从交换集中任取一对染色体011和110,互换它们的第三位,得子孙染色体串010和111,其中111是交换操作创造的新的染色体。因为本例中交换概率为0.5,交换集中剩下二个染色体不再参加交换,而与交换后得到的子孙一起作为下一代种群的成员。由此得到第一次交换操作后第一代种群,如表10-6所示。

表10-6 第1代种群的适应度

| 序号 <i>i</i> | 染色体串 $x_i$ | 适应度 $f(x_i)$ |
|-------------|------------|--------------|
| 1           | 111        | 7            |
| 2           | 010        | 2            |
| 3           | 110        | 6            |
| 4           | 010        | 2            |
| 适应度总和       |            | 17           |
| 最坏适应度       |            | 2            |
| 最好适应度       |            | 7            |
| 平均适应度       |            | 4.25         |

#### (5) 评估新一代的种群的适应度

由表10-6可知,在新一代(第1代)种群中,最优染色体适应度由原来的6提高到了7,其对应的代码串是111,表示一种优化的经营策略,即低价格销售PC机,配套软件表示MS DOS,快速服务,可获最高利润7%(代码串111的适应度)。平均适应度由原来的3提高到4.5,最坏适应度由原来的1提高到2,整个种群的适应度从总体上提高了,被优化了。

遗传算法重复地执行每一代的运算操作,产生新一代,直到满足某些终止标准或条件。在本例中,假定已知染色体数字串111代表的是已知的最好利润7%,则遗传算法停止运行。

例2: 函数优化问题。设有函数  $f(x)=x^2$ ,求其在区间[0,31]的最大值。下面用遗传算法求解这一问题的。

(1) 确定适当的编码,把问题的可能解表示为染色体数字串。因为有一个决策变量  $x$ ,其取值范围为[0,31], $2^5=32$ ,使用5位无符号二进制数组成染色体数字串,即可表达变量  $x$ ,以及问题的解答方案。

(2) 选择初始种群。通过随机的方法产生由4个染色体数字串组成的初始种群,见表10-7

(a) 左端第二列。

表10-7(a) 函数优化过程

| 编号 | 初始种群  | X 值 | 适应度(目标函数) $F(X)=X^2$ | 选择概率 $f_i/\sum f$ | 期望适应度 $f_i/f$ | 实选染色体编号 |
|----|-------|-----|----------------------|-------------------|---------------|---------|
| 1  | 01101 | 13  | 169                  | 0.14              | 0.58          | 1       |
| 2  | 11000 | 24  | 576                  | 0.49              | 1.97          | 2       |
| 3  | 01000 | 8   | 64                   | 0.06              | 0.22          | 0       |
| 4  | 10011 | 19  | 361                  | 0.31              | 1.23          | 1       |
| 和  |       |     | 1170                 | 1                 | 4             | 4       |
| 平均 |       |     | 293                  | 0.25              | 1             | 1       |
| 最大 |       |     | 576                  | 0.49              | 1.97          | 2       |

表10-7(b) 函数优化过程

| 复制后交换集种群 | 交换配对 | 交换位置 | 新种群   | X 值 | $F(X)=X^2$ |
|----------|------|------|-------|-----|------------|
| 0110 1   | 2    | 4    | 01100 | 12  | 144        |
| 1100 0   | 1    | 4    | 11001 | 25  | 625        |
| 11 000   | 4    | 2    | 11011 | 27  | 729        |
| 10 011   | 3    | 2    | 10000 | 16  | 256        |
| 和        |      |      |       |     | 1754       |
| 平均       |      |      |       |     | 439        |
| 最大       |      |      |       |     | 729        |

(3) 计算适应度值及选择概率。此问题中染色体的适应度取函数自身  $f(x)=x^2$ 。为了计算每个染色体的适应度,先将染色体解码,求出其二进数字串等价的十进制数,即  $x$  值,再由  $x$  值计算目标函数  $f(x)=x^2$  的值,作为适应度值。在此基础上计算选择概率  $P_i=f_i/\sum f$ , 适应度期望值  $f_i/f$ ,计算结果如表10-7(a)第5列、第6列所示。

(4) 选择进入交换集的染色体。按适应度比例法,选择进入交换集的染色体串,如表10-7(b)第1列所示。可见,染色体1和4均被选择了1次;染色体2被选择了1次;染色体3没有被选择,也就是说染色体3被淘汰了。所选择的4个染色体被送到交换集,准备参加交换。

(5) 交换染色体。首先对进入交换集的染色体进行随机配对,此例中是染色体2和1配对,染色体4和3配对。接着随机确定交换位置,结果是第1对染色体的交换位置是4,第2对染色体的交换位置是4。经交换操作后得到的新种群如表10-7(b)第4列所示。

(6) 变异。此例中变异概率取为0.001。由于种群中4个染色体总共有20位代码,变异的期望值为  $20 \times 0.001 = 0.02$  位,这意味着本群体中无法进行变异。因此,此例中没有进行变异操作。

从表10-7可以看出,虽然仅进行了一代遗传操作,但种群适应度的平均值及最大值却比初始种群有了很大提高,平均值由293变到439,最大值由576变到729。这说明随着遗传运算的进行,种群正向着优化的方向发展。

### 第三节 图式定理

由前面的叙述可知,遗传算法并不复杂,但却展示了强大的信息处理能力,为什么它具有这种能力呢? Holland 提出的图式定理(Schmata)在一定程度上对此作了解释,奠定了遗传算法的理论基础。图式是描述种群中任意染色体之间相似性的一维符号串。它定义在符号表0,1,\*之上,即由二进制数字0,1及通配符\*任意组合而成。图式中的01序列组成其固定部分,\*序列表示其变化部分,整个图式表示有意义的匹配模式。例如,图式01\*\*可和0100,0101,0110,0111中的任一个匹配。定义在0,1,\*之上且长度为L的符号串所能组成的最大图式数为 $(2+1)^L$ ,若 $L=4$ ,则它最多可组成27种图式。含有N个染色体的种群可能包含的图式数在 $2^L \sim N \cdot 2^L$ 之间。遗传算法正是利用种群中包含的众多的图式及其染色体符号串之间的相似性信息进行启发式搜索和问题求解。已经证明,在产生新一代的过程中,尽管遗传算法只完成了正比于种群长度N的计算量,而处理的图式数却正比于种群长度N的三次方,这就是遗传算法隐含的并行机制。

用两个参数描述图式H,即:

- 定义长度  $\delta(H)$ ——H左右两端有定义位置之间的距离。
- 确定位数  $O(H)$ ——H中有定义(非\*)位的个数。例如,图式1\*\*\*\*的定义长度是5—1=4,确定位数为2。

种群中定义长度短,确定位数少,适应度高的的图式(符号串)称为组块(building block)。GA的运算实际上是对组块的操作。图式定理可用来描述即图式的组块在多大程度上能保留到下一代,详述如下:

设  $m(H,t)$  表示在第t代种群中存在的图式H的数量,  $f(H)$  为包含H的染色体平均适应度, FAV 为种群中所有染色体的平均适应度, 在经过复制操作后,  $t+1$  代 H 的数量为:

$$m(H,t+1) = m(H,t) \frac{f(H)}{FAV} \quad (10-1)$$

由上式可知,在遗传算法的运算过程中,下一代种群中H的数量正比于H所在染色体平均适应度与种群平均适应度的比值。若某些染色体数字串的适应度高于当前种群中平均适应度时,这些数字串所包含的组块在下一代数字串中出现的机会将增大,否则在下一代中组块的出现将减少。运算中组块的增减是并行进行的,这也表现了遗传算法隐含的并行性。

经过交换和变异操作后,图式可能被破坏,因交换而破坏的概率为:

$$\frac{P_e \cdot \delta(H)}{L-1}$$

式中: L 为染色体长度,  $P_e$  为交换的概率, 即在经选择操作进入交换集的一对染色体上发生交换的概率。

因变异操作而破坏的概率为:

$$O(H)P_m$$

式中:  $P_m$  是变异概率,  $O(H)$  是确定位数。

综合上述讨论,在GA运算过程中,经过复制,交换和变异操作后,  $t+1$  代图式H的数量为:

$$m(H,t+1) \geq m(H,t) \frac{f(H)}{FAV} [1 - P_e \frac{\delta(H)}{L-1} - O(H)P_m] \quad (10-2)$$

式(10-2)也可用文字表达为如下的图式定理：

在选择、交换、变异运算的共同作用下,确定位数少、定义长度短、适应度高于平均适应度的图式,在下一代中以指数级增长。由此定理可知,下一代图式(组块)的数量正比于染色体数字串的适应度。为了增加图式,就应尽可能减少交换和变异。但是,没有交换和变异,就没有种群的进化,就不能跳出可能陷入的局部最小区域。遗传算法正是通过组块的增减、组合等处理来存优去劣,寻找最佳匹配点,进行高质量的问题求解。

## 第四节 遗传算法应用中的一些基本问题

### 一、知识表示(编码)

知识表示要解决的问题是如何用方便实用的染色体串表达欲求解问题的候选解,即如何编码问题的候选解集。编码的恰当与否对问题求解的质量和速度有直接的影响。David E.-Goldberg 提出了编码的两条基本原理:

- (1)所选编码方式应能容易地生成与求解问题有关的确定位数少、定义长度短的组块。
- (2)所选编码方式应具有最小的字符集,自然地表达欲求解的问题。

原理(1)基于图式定理,在实际编码过程中较难掌握、使用,原理(2)则为实际编码工作指出了方向。根据原理(2),由于二进制编码的字符集小,它比非二进制编码要好。这也可从以下数学推演来说明。设某问题的解既可以用  $b$  位二进制数,也可以用基数为  $k$ ( $k$  进制)的  $d$  位非二进制数编码,为了保证这两种编码方式确定的解的数量相同,应有:

$$2^b = k^d$$

二进制数编码时每个码位的取值有 0,1,“三种情况,可得到的图式数为  $3^b$ 。 $k$  进制数编码时每个码位的取值有  $k-1$  种情况,可得到的图式数为  $(k+1)^d$ 。由于  $b > d$ (以十进制为例, $b \approx 3.33d$ ,即,一位十进制数若用二进制数表达,约需 3.3 位), $3^b > (k+1)^d$ ,也就是说采用二进制数能比非二进制数提供更多的图式。

从另外一个角度看,由于实际问题中往往采用十进数,用二进制数字串编码时,需要把实际问题对应的十进数变换为二进数,使其数字长度扩大约 3.3 倍,因而对问题的描述更加细致,而且加大了搜索范围,使之能够以较大的概率收敛到全局解;另外,进行变异运算时工作量小(只有 0 变 1 或 1 变 0 的操作)。所以,早期遗传算法的编码多采用二进制数。

但是,采用二进制编码也有一些缺点,如:编码时需进行十进制数到二进制数变换,输出结果时要解码,进行二进制数到十进制数的变换;当二进数串很长时,交换操作计算量很大。也有人认为,使用二进制编码效率低,可能会使遗传算法的性能变坏。因此,近年来许多应用系统已使用非二进制编码,如 Gencop 系统采用的十进制数编码,Samel 系统采用的符号(字符串)编码等。十进制串编码与人的自然习惯比较吻合,不需要进行十一二进制数之间的变换;由于数字串长度比二进制数短,交换运算计算量比二进制小;但变异运算却比二进制编码复杂(各个数位变异的可能性有九种)。总之,这两种编码方法各有优缺点,到底采用哪种方法最好,还须在理论上作进一步的深入探讨,具体应用时,可根据所求解问题的性质灵活决定。

## 二、适应度函数

在一般的遗传算法中,基本不用搜索空间的知识,而仅用适应度函数来评价染色体的优劣,并在此基础上进行各种遗传操作,因此适应度函数选择是否适当对算法的性能好坏影响很大,应根据实际问题的特性具体确定。通常遗传算法要求适应度函数值非负,同时要求把待解优化问题表达为最大化问题,即目标函数的优化方向对应适应度函数的增大方向,而一般的优化问题并不一定满足这些条件。许多问题求解的目标是求目标函数(如成本和费用)的极小值或者负值。有些情况下,如最优排列顺序问题的求解根本没有目标函数,而只有优化要求。另外,若种群规模不大,而在遗传算法运行的开始阶段,有少数适应度极高的染色体,则按通常的选择法,其选择概率很高,这些染色体就会大量繁殖(复制),在种群中占有很大比例,减少了种群的多样性,导致算法过早收敛,可能丢失了一些最优点,而陷入局部最优点。在算法的结束阶段,若大多数染色体的适应度都很高,使种群的最高适应度和平均适应度相差不大,在平均适应度附近和具有最高适应度的染色体被选中的机会基本相同,选择就会成为近乎随机的过程,适应度的作用就会消失,算法基本上限于停顿状态。为了使遗传算法能正常运行,同时保持种群内染色体的多样性,改善染色体适应度值的分散程度,使之既要有差距,又不要差距过大,以利染色体之间的竞争,保证遗传算法的良好性能,需对所选择的适应度函数进行某些数学变换。几种常见的变换方法如下。

(1) 非负变换:把最小化优化目标函数变换为以最大值为目标的适应度函数:

$$f(z) = \begin{cases} c_{\max} - z & z < c_{\max} \\ 0 & \text{其他情况} \end{cases}$$

式中:  $z = g(z)$  是适应度函数;  $c_{\max}$  是常数,通常取为迄今为止进化过程中  $z$  的最大值,或者为了保证  $f(z)$  为正而预先设定的一个与种群无关的常数。

(2) 线性变换: 把优化目标函数变换为适应度函数的线性函数。

$$f(z) = az + b$$

$a, b$  是系数,可根据具体问题的特点和期望的适应值的分散程度在算法开始时确定或在每一代过程中重新计算。

(3) 幂变换: 把优化目标函数变换为适应度函数的幂函数。

$$f(z) = z^\alpha$$

式中:  $\alpha$  是常数,据经验确定。Gillies(1985年)在机器视觉应用中取  $\alpha=1.005$ 。

(4) 指数变换: 把优化目标函数变换为适应度函数的指数函数。

$$f(z) = e^{(-\beta z)}$$

式中:  $\beta$  是常数。这种变换已用于机器人应用中。

## 三、控制参数和终止判据

控制遗传算法运行的主要参数是种群长度,染色体长度,遗传操作概率(交换概率  $P_c$  和变异概率  $P_m$  等,这些参数的设定目前尚无统一的理论指导,主要根据具体问题和实验确定。例如从维持种群中染色体的多样性以避免陷入局部最优方面考虑,种群长度越大越好,但这会增加计算量并对染色体间的竞争有不利影响,究竟选择多长要根据软硬件环境及具体问题确定。

遗传操作概率,特别是变异概率的选择更为困难,选择不当便会产生初始收敛而陷入局部最优。一般地,选择交换概率为0.6~0.9,变异概率为0.001~0.01。

由于遗传算法没有利用目标函数的梯度等信息,无法确定染色体在解空间中的位置,无法用常规数学方法确定遗传算法是否收敛。有人已经证明,在任意初始化、任意交换算子以及任意适应度函数下,遗传算法都不能收敛到全局最优解。而通过改进遗传算法,即不按比例法进行选择,而是在选择作用前(或后)保留当前所得最优解,则能收敛至全局最优解。但是收敛到最优解的时间可能非常长。

综上所述,目前使用严格的数学方法判定遗传算法的终止(收敛)条件还比较困难。因此,大多数应用系统在实际中采用的主要是启发式方法,如用以下条件来判定算法的终止:

- 是否到了预定算法的最大代数;
- 是否找到某个较优的染色体;
- 连续几次迭代后得到的解群中最好解是否变化等。

## 第五节 高级遗传算法(RGA)

前面介绍的遗传算法仅包含选择、交换、变异三种遗传子,其交换及变异概率是固定不变的,这样的遗传算法被称为简单(经典)遗传算法(SGA)。随着遗传算法研究的进展,人们在SGA的基础上提出了多种多样的遗传运算和选择方法,从而形成了各种较为复杂的遗传算法,称为高级遗传算法(Refine Genetic Algorithms,简称为 RGA),其基本特征是含有多种遗传运算,交换及变异概率可变,因而具有更强的信息处理和自适应能力。

### 一、改进的选择方法

前面介绍了在SGA中使用的选择方法—适应度比例法,此法的优点是实现简单,缺点是选择过程中最好的染色体可能在下一代产生不了子孙,可能发生随机错误。下面介绍一些改进的选择方法。

#### (1) 择优选择法

对群体中最优秀(适应度最高)的染色体不进行交换和变异运算,而是直接把它们复制到下一代。以免交换和变异运算破坏种群中的优秀解答。这种方法可加快局部搜索速度,但又可能因群体中优秀染色体的急剧增加而导致搜索陷入局部最优解。

#### (2) 确定性选择法(期望值法)

先计算群体中每个串的生存概率

$$P_j = f_j / \sum f_i, \quad 1 \leq j \leq n$$

然后计算期望复制数

$$e_i = P_j \cdot n$$

式中:  $n$  为群体中染色体的数目。根据  $e_i$  值的整数部分给每个染色体串分配一个复制数,而按  $e_i$  的小数部分对群体中的染色体串排序,最后按排列的大小顺序选择要保留到下一代的染色体串(子孙)。

#### (3) 有替换随机剩余选择法

此选择法的开始与确定性选择法相同,在计算出期望复制数  $e_i$  值后,把整数部分赋给对

应染色体串，但其小数部分作为适应度比例法的权，再按适应度比例法进行选择。

#### (4) 无替换随机剩余选择法

此方法与有替换随机剩余选择法的区别在于：把  $e_i$  值的小数部分看作概率，然后挑选生存概率等于  $e_i$  的小数部分的其他染色体串作为复制的下一代的染色体串。例如，期望复制数等于 1.5 的一个染色体会有两个子孙，其中一个的  $e_i$  值等于 1，另一个  $e_i$  值等于 0.5。

#### (5) 顺序选择法

根据各染色体适应度值由大到小进行排序，再把预先设计的一组选择概率值按大小顺序分配给各染色体，最后按选择概率来确定要选择的子孙。

## 二、高级遗传运算和算法

### 1. 多点交换(multiple-point crossover)

交换运算的目的是把欲交换两个染色体中性能优良的组块，遗传到下一代某个染色体中，使之具有父辈染色体的优良性能。但是，在某些情况下，一点交换运算无法达到这个目的。例如，设有两个父辈染色体  $c_1$  及  $c_2$ ：

$c_1$ : 10110001100

$c_2$ : 00101101001

设  $c_1$  中含有性能优良的组块 1\*1 及 \*\*0， $c_2$  中含有性能优良的组块 1\*1 及 1\*\*1，若对  $c_1$  和  $c_2$  进行一点交换运算，则无论交换点选在何处，都可能使这些优良组块由于交换运算而被分割或丢弃，不能遗传到下一代。而采用二点交换就可避免这个问题。设二个交换点选择如下图(图中'表示交换点)：

$c_1$ : 1011| 0001| 100

$c_2$ : 0010| 1101| 001

则二点交换运算就是交换  $c_1$  与二交换点之间的部分，得到两个子孙染色体  $c'_1$  及  $c'_2$  如下：

$c'_1$ : 1011| 1101| 100

$c'_2$ : 0010| 0001| 001

可见，子孙染色体  $c'_1$  中继承了父辈染色体中性能优良的组块。从这个例子中也可以看到多点交换的优越性。

多点交换是对一点交换的拓广，其操作与一点交换类似，也是先确定交换点数，随机选择交换点(位置)，再进行交换操作。若取交换点数  $N_c$  为 1，多点交换就变成了一点交换。若  $N_c$  为偶数，可视染色体为一个基因头尾相接组成的环，绕着环匀速前进，随机地选择交换点，再交换两个父辈染色体中由交换点所确定的弧段。 $N_c$  为奇数时，取缺省交换点为染色体串的位置 0 (即染色体的头部)，则它可等效为偶数点交换的情况。

### 2. 一致交换(uniform crossover)

这是多点交换的另一种实现方式，由 Syswerda 1989 年提出，它使用随机产生的、长度与父辈染色体相同的二进制掩码决定父辈染色体中交换的对应位，掩码为 0 的位交换，掩码为 1 的位不交换(这个交换规则也可以相反，即掩码为 1 的位交换)。例如：

父辈染色体  $S_1$ : 0 1 1 0 0 1 1 1

父辈染色体  $S_2$ : 1 1 0 1 0 0 0 1

掩码: 0 1 1 0 1 0 0 1

子孙染色体  $S_1'$ : 1 1 1 1 0 0 0 1

子孙染色体  $S_2'$ : 0 1 0 0 0 1 1 1

从原理上讲,一致交换可生成任意新的组块,有利于搜索到解空间中新的区域,但也有可能因进行一致交换而破坏父辈染色体中性能优良组块的遗传。

### 3. 部分匹配交换运算(PMX)

这是 Goldberg 等人在求解 TSP(旅行商)问题时提出的一种交换方法。它先用随机均匀分布方法在欲交换两父染色体串中各产生两个交换点,把这两点之间的区域定义为匹配区域,再对两个匹配区域中的基因通过对应位匹配置换。例如,两父染色体串为:

$E_1$ : 9 8 4 \*O5 6 7 \*O 1 3 2 10

$E_2$ : 8 7 1 \*O4 10 3 \*O 2 9 6 5

符号“\*O”表示交换点。

对染色体串  $E_1$ ,其匹配区域中的三个元素“5 6 7”应与染色体串  $E_2$ 匹配区域中的三个元素“4 10 3”逐一匹配,即通过在  $E_1$ 内元素的置换,使匹配区域内这三个元素换为“4 10 3”。为此, $E_1$ 中的元素5与4,6与10,7与3分别换位,得到  $E_1'$ 如下:

$E_1'$ : 9 8 5 \*O4 10 3 \*O1 7 2 6

对染色体串  $E_2$ ,其匹配区域中的三个元素“4 10 3”应与染色体串  $E_1$ 匹配区域中的三个元素“5 6 7”逐一匹配,通过在  $E_2$ 内元素的相应置换,得到  $E_2'$ 如下:

$E_2'$ : 8 3 1 \*O5 6 7 \*O 2 9 10 4

### 4. 重组运算(reordering operators)

在有些情况下,染色体的特性不仅与基因的位值有关,而且与基因在染色体中的位置有关,另一方面,某些性能优良且密切相关的基因在染色体中的位置相隔较远,通过重新对染色体进行排列,就可把这些基因组合在一起,以提高优良组块的在种群中的繁殖率。重组就是对染色体中的基因进行重新组合的运算。通常,重组运算可通过倒位操作来实现。进行倒位操作时,沿染色体长度方向选择二点,把这二点之间的基因位置颠倒过来,形成一个新的染色体。例如,染色体 C 及其各基因的位置编号如下:

C 中各基因的位置编号: 1 2 3 4 5 6 7 8

C: 0 1 1 1 1 0 1 0

随机选择二个倒位点(以“\*O”表示):

1 2 \*O3 4 5 6 \*O7 8

0 1 \*O1 1 1 0 \*O1 0

颠倒两个 \*O 之间基因的位置,得到新染色体 C'

C' 中各基因的位置编号: 1 2 6 5 4 3 7 8

C': 0 1 0 1 1 1 1 0

## 三、混合遗传算法

由前面的介绍可知,遗传算法在优化方面具有实用性和稳健性。但是,由于遗传算法在局部搜索方面功能的不足,对特定领域问题来说,遗传算法一般不是最成功的优化算法。解决这个问题的途径有两条,一是对遗传算法进行进一步的修改,二是把遗传算法与现有优化算法(如梯度下降法、黄金分割法、启发式搜索算法、模拟退火法、列表寻优法等)相结合,互相取长

补短。例如,组合优化计算中贪婪(Greedy)算法是一种有效的局部搜索方法,Louis Anthony COX等把GA与贪婪算法相结合,求解实际电信网络的布线问题,取得很好的效果。David J.-Powell等混合使用遗传算法、专家系统与数学优化算法求解工程设计优化问题,获得成功。许多研究者的实践表明,遗传算法与现有优化算法的结合,可产生比单独使用遗传算法或现有优化算法更好,更实用的算法。

Davis, Lawrence 1991年在《遗传算法手册》中指出了遗传算法与现有优化算法(以下简称现有算法)结合的几点原则,现归纳于下:

#### (1)采用现有算法的编码

这样做好处有两个:一是嵌入在现有算法编码中的领域专家知识可以在混合遗传算法中继续使用;二是由于混合遗传算法与现有优化算法将在相同的环境下运行,使实际应用人员感觉混合遗传算法的使用比较自然。

#### (2)吸取现有算法的精华

·若现有算法是快速的,则把它产生的解答集添加到混合遗传算法的初始解群中,在此基础上进行各种遗传运算。这样,由混合遗传算法产生的优化解答至少不会比现有算法差。

·若现有算法对编码进行一系列变换,把这些变化合并到混合遗传算法的运算操作中是非常有用的。

·若现有算法对其编码的解释是清楚的,可把其译码技术合并到混合遗传算法的解码及适应度评价操作中,以节省译码计算时间。

#### (3)改进遗传运算

混合现有算法的目标是具有遗传算法和现有算法二者的优点,但是,在现有算法的编码方式(一般是实数编码)后,遗传算法原有的那些基于二进制编码的运算(如交换、变异等)就可能不再适合了,因此,必须根据所求解问题的性质对原有的遗传运算进行改进。

## 四、并行遗传算法

GA固有的并行性使它比较适合并行计算,但Holland经典的GA是不能直接并行执行的,这是因为在GA执行时要考虑全局控制,如在选择和重组(交换和变异)阶段,需要串行地执行一些代码,并迫使在处理机之间进行非局部的相互作用,因而需要很高的通信开销。为了克服经典GA这方面的缺陷,人们提出了许多并行GA策略和模型,下面介绍两种典型的并行GA模型。

### 1. 邻域(细粒度)模型(Neighbourhood Model-NM)

NM模型的出发点是开发一个群体内的并行性。在此模型中,只有单个群体在进化,每个个体被放置在平面网格的细胞内,选择和重组只在网格中相邻染色体之间进行。NM模型的基本算法如下。

(1)适当定义欲求解问题的表示;随机产生候选解的群体,并把它划分为N个子群体;定义各子群体的邻域结构。

(2) FOR 每个子群体 SP<sub>i</sub> (i=1,2,...,N) REPEAT

P(0)=(P<sub>1</sub>,...,P<sub>n</sub>) 初始化

WHILE NOT(终止条件)

P(t)→P(t+1) 选择复制

$P(t+1) \rightarrow P'(t+1)$  交换

$P'(t+1) \rightarrow P''(t+1)$  变异

$P(t+1) = P''(t+1)$

$t = t + 1$

(3) 送最好的染色体到  $n$  个邻域, 接收邻域最好的染色体来替换本群体中最坏的染色体。

## 2. 岛屿(粗粒度)模型(Island Model, LM)

LM 模型的出发点是开发群体之间的并行性。在此模型中, 把群体细分为若干个互相隔离的子群体, 分配到各个处理机; 各处理机独立地进行子群体的各种遗传操作, 并行地进化, 并周期性地把最好的染色体迁移到相邻的子群体中。LM 模型的基本算法如下。

(1) 随机产生候选解的初始群体: 把每个染色体赋给网格的一个元素; 计算每个染色体的适应度;

(2) 对网格的每个当前元素  $C$  执行步骤(3)(4)(5)(6);

(3) 在赋给  $C$  及其邻域的染色体中选择一个新染色体占据  $C$ ;

(4) 在  $C$  的邻域中随机选择一个染色体, 与  $C$  中的染色体以概率  $P_e$  进行交换(重新组合);

(5) 以概率  $P_m$  对  $C$  中的染色体进行变异;

(6) 计算赋给  $C$  的新染色体的适应度;

(7) 若不满足结束条件, 转步骤(2)执行。

上述两种模型可自然地用不同类型的并行计算机实现。一般来说, LM 模型适于 MIMD(多指令流多数据流)计算机, 如基于 TRANSPUTER 的并行计算机实现; NM 模型适于 SIMD(单指令流多数据流)计算机, 阵列机或连接机等并行计算机实现, 但是也有例外情况。一个典型的例子是 TANESE 的工作(1987年), 他在由 NCUBE 公司制造的一台超立方体 MIMD 计算机上实现了 NM 模型。该计算机由64个微处理机互联组成了一个6维的二进制立方体。立方体共有  $2^6 = 64$  个结点, 每个结点是一台有32位字长的 CPU 和128K 字节局部存储器的微处理机, 其存储器的四分之一用来存储结点操作系统和信息缓存器。每个微处理机运行一个子群体, 与其邻域(直接相联)的其他6个微处理机周期性地交换最好的染色体。另一个例子是 MUHLENBEIN 的 ASPARAGOS 系统(1988年): 此系统的邻域结构由连接成梯状的两个环组成, 直接映射到由64个 TRANSPUTER 组成的 MIMD 计算机实现。实验结构表明, 这两个系统所实现的并行 GA 模型都获得了很好的效果。

## 第六节 遗传算法(GA)应用举例

GA 的早期应用主要围绕组合优化问题求解, 如煤气管道的最优控制, 旅行商(TSP)问题等, 近年来迅速扩展到机器学习、设计规划、神经网络优化、核反应堆控制、喷气发动机设计、通信网络设计、人工生命等领域, 显示了 GA 应用的巨大潜力。

常规的数学优化技术基于梯度寻优技术, 计算速度快, 但不足之处是: 要求其目标函数可微; 常因陷入局部解而不能获得最优的全局解。工程中存在许多困难的组合优化问题及复杂的函数优化问题, 大都具有非线性, 有些甚至不可微, 对这类问题常规的数学优化技术无法有效的求解, 只能对问题作简化的近似处理。GA 本质上是一种基于随机的全局多点搜索算法, 求解优化问题时, 具有以下特点: 只使用适应度函数反复指导和执行遗传搜索, 不需要对目标函

数求导数,因而对目标函数没有可微的要求;同时搜索解空间中的许多点,而不是一个点,保证了解的全局最优性;使用概率而不是确定性规则指导搜索,能搜索离散的多峰复杂解空间。因此,GA 特别适合求解具有非线性,不连续,不可微,多峰的目标函数,或约束条件复杂的非线性组合优化问题。下面举例说明 GA 在组合优化问题求解中的应用。

## 一、GA 在 TSP(旅行商)问题求解中的应用

设存在  $N$  个城市,  $D_{ij}$  表示城  $i$  与城  $j$  之间的距离,  $D_{ij} = D_{ji}$ , 现在要求一条遍历所有  $N$  个城市,且不走重复路的最短路径(最短哈密尔顿圈)。

这是一个典型的组合排列顺序问题,属 NP 难题。为求解方便,采用十进制编码,每个染色体由按一定顺序排列的  $N$  个城市的序号组成,表示一条可能的旅行路径,其中每个基因表示一个城市。染色体的长度为  $N$ ,解空间为  $2^{N-1}$ 。适应度取为一条旅行路径对应的距离,路径越短的染色体适应度越高。例如,取  $N=7$ ,城市代号为 1,2,3,4,5,6,7,则种群中的染色体

1 3 2 4 5 6 7

表示一条旅行路径:从 1 → 3 → 2 → 4 → 5 → 6 → 7 → 1

其适应度为  $\sum D_{ij} = D_{13} + D_{32} + D_{24} + D_{45} + D_{56} + D_{67} + D_{71}$

下面介绍具体算法:

- (1) 确定种群长度、城市数目、遗传代数、变异概率等参数;
- (2) 计算  $D_{ij}, i, j = 1, 2, \dots, N$ , 并随机组合成染色体,生成初始种群;
- (3) 计算每个染色体的适应度:  $\sum D_{ij}$ ;
- (4) 计算选择概率、交叉概率;按轮转法进行选择,复制路径较短的染色体到匹配集;删除路径较长的染色体;
- (5) 随机产生交叉点,进行交叉操作,交叉方法与二进制编码时类似。

(6) 进行变异操作。由于采用的是十进制编码,变异操作与二进制编码时不同,将被变异染色体中的某些城市次序颠倒,即互换两个城市的位置。

(7) 评价适应度,检查终止条件。若满足终止条件,算法停止,否则转到(3)继续算法过程。

当前,用 GA 求解 TSP 问题的研究已取得大量理论和实际成果,对 431 个城市的 TSP 问题已取得了最优解,对 666 个城市可取得准最优解,其方法也拓展到了工厂生产调度领域。实践证明,用 GA 求解 TSP 问题的在运算速度及解的质量上明显优于其他方法,如传统解法和 HOPFIELD 神经网络法。

## 二、GA 在电网优化规划中的应用

电网规划的任务是在已知规划水平年负荷和电源规划的基础上,根据现有网络结构和待选线路,确定出满足运行要求且最经济的网络规划方案。

### (1) 编码

采用二进制编码。每个染色体表示一个规划方案,其中每个基因表示一条待选线路,当某基因为 1 时,表示其对应的线路被选中加入了网络,为 0 表示该线路未选中。例如,当染色体长度为 6 时,表示有 6 条待选线路,染色体 100110 表示该方案将第 1,4,5 条待选线路加入了系统。

## (2)适应度函数

根据电网规划要求投资最小且不出现过负荷的目标,构造如下的适应度函数:

$$f(z)=\begin{cases} U_0 - c(z) & c(z) \leq U_0 \\ 0 & c(z) > U_0 \end{cases}$$

$$c(z) = \sum c_i \times Z_i + p \times w$$

式中: $U_0$ 是给定的常数; $c(z)$ 为规划方案的费用; $\sum c_i$ 为所有为待选线路*i*的建设费用; $Z_i$ 为染色体中第*i*个基因的值; $w$ 为网络的过负荷量; $p$ 是出现过负荷量时的惩罚系数。

## (3)计算过程

计算过程与前例基本类似,但是根据本问题的特点,把变异操作改成了成对变异。根据Holland的基本定义,变异是将某个基因由0变1或1变0,在电网规划中这样做意味着增加或减少一条待选线路,使规划的品质发生变化,因而是行不通的。成对变异的思想是当某个基因由1变0,即去掉一条线路时,设法寻找另一条更经济的线路加入系统,即把费用更低的线路对应的基因位由0变1。

研究实验结果表明,应用GA解决电网优化规划问题,可克服一般优化算法的局部最优化及维数灾难,能在较短时间内提供一批优化方案,具有很好的应用前景。

## 三、GA 在木材切割优化中的应用

通常木材的销售价格取决于其质量等级及长度,1级价最高,4级价最低,每平方米木材的价格随其长度而增加。木材的销售可整块,也可切成数段进行,以求最高利润。若木材最多可切成*N*段,则切割方案共有 $2^{N-1}$ 种组合,即问题的解空间为 $2^{N-1}$ 。某木材公司销售的木材长20米,要求最大切割段数为10,每段的长度最少2米,求可获最高售价的切割方案。显然,此问题的解空间为 $2^9$ ,在这样大的搜索空间内由人工判决那种方案最优(利润最高),几乎是不可能的。下面用GA求解这个问题。

根据问题的性质,采用二进制编码表达知识。首先在20米长的木材上每2米打一个分段标记,再用1位二进制数表示在每个切割位置是否切割,1表示切,0表示不切,最后把九个切割位置对应的二进制数连接成一个9位的染色体数字串,每个数字串表示一种切割方案。例如,数字串100001011表示在第1,6,8,9切割位置进行切割,经解码可输出两个2米的2级板,一个4米的1级板,一个10米的4级板,一个2米的3级板。染色体的适应度取为经切割后木材的售价,价格越高,适应度值越大。系统的控制参数为:种群长度20;染色体长度9,最大代数20,交叉概率0.6,变异概率0.033。采用的选择,交叉及变异操作与第三节所述基本相同。算法经过20代的进化运算后,获得了最优解。

## 习 题 十

1. 简述遗传算法的工作原理和过程。

2. 解释名词

(1)轮转法;

(2)交换;

• 218 •

(3) 变异。

3. 图式定理的中心思想是什么,其意义何在?
4. 遗传算法怎样体现其隐含的并行性?
5. 与常规数学优化技术(如梯度寻优技术等)相比,遗传算法有什么优越性?
6. 举例说明遗传算法可用于解决哪些工程问题。

## 第十一章 专家系统概述

### 第一节 专家系统简介

#### 一、专家系统概念及发展动态

专家系统(ES)是一种大型复杂的智能计算机软件,是人工智能开始走向实用化的标志和里程碑,是人工智能从一般思维规律探索走向专门知识利用的突破口。它把专门领域中若干个人类专家的知识和思考、解决问题的方法以适当方式储存在计算机中,使计算机能在推理机的控制下模仿人类专家去解决问题,在一定范围内取代专家或起专家助手作用。可以说,专家系统是专门领域中的知识工程系统。自从60年代中期首批在美国斯坦福大学和麻省理工学院问世以来,专家系统技术迅猛发展,尤其是70年代中期以来,各种实用专家系统不断涌现,广泛用于化学、数学、医疗、地质、气象、石油勘探、军事、法律、教育等领域,取得了重大的社会效益。1986年举行的美国人工智能学会第六届年会(AAAI—86)有5100人参加,两年后举行的AAAI—88出席人数达到6000。这两次会议是对人工智能,特别是对专家系统应用成果的大检阅。会议认为,人工智能不仅作为未来计算机系统的一种技术,而且正在扎实地走向实用化。人工智能技术已走出研究实验室而进入一般机关、企业应用,美国大多数较大的组织正在准备或已经实施专家系统计划。截止到1988年,全世界已有2000个专家系统在使用,8500个专家系统正在开发中。IBM公司内部有100个专家系统正在使用。IBM公司还成立了人工智能项目办公室,在普林斯顿建立了专家系统销售中心,在麻萨诸塞建立了人工智能支持中心,并投资870万美元,与33所大学开发52个人工智能研究项目。DEC公司有700名技术人员从事人工智能研究,并从1979年就使用了计算机配置,计算机设计专家系统。据报道,DEC公司的计算机配置专家系统XCON开始每年为该公司节省1500万美元的开支,每年能获得1.5亿美元的利润。不仅工业界,各发达国家的政府也大力支持专家系统的开发研究工作。日本曾投资4亿5千万日元研究以专家系统技术为核心的第五代计算机;美国国防部也提供了6亿美元研究第五代计算机;英国则相应制定了阿尔维计划;我国政府1986年3月制定的863高技术发展计划也以人工智能,专家系统技术为基础。1995年我国制定的九五计算机技术科技攻关规划建议仍把人工智能技术作为四个重点发展的关键技术之一,鼓励继续开发各种实用专家系统及其开发工具。1995年5月,第一届亚洲太平洋地区专家系统国际会议在我国黄山举行,对国际国内专家系统技术的发展起到了进一步的促进作用。

#### 二、专家系统的特点

专家系统之所以受到欢迎和重视,是因为它具有下列主要特点:

(1)可存储一个或多个专家的知识和经验,能以接近专家的水平在特定领域内工作。

- (2)能高效、准确、迅速地工作,不会像人类专家那样产生疲倦和不稳定性。
  - (3)使人类专家的领域知识突破了时间和空间的限制,ES 程序可永久保存,并复制任意多的副本,以在不同地区和部门使用。
  - (4)可通过符号处理进行各种形式的推理,也可以对不确定数据进行推理。
  - (5)具有透明性,能以可理解的方式解释推理过程。
  - (6)具有自学习能力,可总结规律,不断扩充和完善系统自身。
  - (7)能提高生产率,产生巨大的社会效益、经济效益、军事效益。
- 在下列情况下,特别需要开发和应用专家系统:
- 用专家系统来工作效率很高;
  - 人类专家的知识很快就要失传,必须通过专家系统来收集、管理和运用;
  - 人类专家太少,必须建造专家系统来使专家们的知识同时用于不同地点的多个岗位;
  - 危险环境需要人类专家,但若让专家亲临该环境保障费用太高。

### 三、专家系统应用举例

由前面的介绍可知,专家系统应用已经相当广泛和深入,要系统、完整地叙述它是非常困难的。下面从专家系统所求解问题的一些典型领域来介绍其应用概况。

#### 1. 专家系统在数据解释方面的应用

数据解释是指通过分析来自测试仪器及传感器的数据,推出系统的状态,给出其物理解释。如声音数据的解释、图像的解释、谱数据的解释等,其本质是按一定的规则(专家知识)对数据进行分类。

美国斯坦福大学开发的 DENDRAL 专家系统可分析由质谱仪获得的数据,鉴别有机化合物的结构。其工作过程由三个阶段组成:在计划阶段,使用由质谱推断未知物质的规则构成能提取未知物质化学结构的约束条件;在生成阶段,以无环式结构作为对象自动生成能满足上述条件的化学结构;在验证阶段,采用由结构式推断谱的规则,对所生成的结构式(代表某种物质),进行有效性评价。Smith 开发的 CONGEN 专家系统可用来鉴别更一般的含有环式结构的有机化合物;Engelmore 的 CRYSTALIS 专家系统把 X 射线绕射形成的电子密度图作为输入,进行蛋白晶体结构鉴别。

#### 2. 专家系统在诊断方面的应用

诊断是指利用因果关系知识对所获取(观测)到的数据进行分类,分析判断对象的故障所在,并给出排除故障的对策。这类系统可分为生物(医疗)诊断(BDIAES)和设备诊断(DDIAES)专家系统两大类型。BDIAES 的第一个典型代表是斯坦福大学开发的 MYCIN 专家系统,它诊断血液细菌感染疾病,并开出适当的处方。基于 MYCIN 的肺功能诊断系统 PUFF 已在美国旧金山附近的太平洋医疗中心日常门诊使用。匹兹堡大学 1977 开发成功并不断改进的 QME 专家系统是当今世界上规模最大的疾病诊断专家系统,现在该系统已经包含 75 条医疗知识,可诊断 600 种不同类型的疾病。DDIAES 的一个典型是贝尔研究所开发的电话线路故障诊断系统 ACE。它连接在线路维护用数据库系统 CRAS 上,接收 CRAS 的数据,对电话线路进行故障诊断,输出线路故障报告书。

#### 3. 专家系统在监测方面的应用

监测是指根据联机实时系统的数据,检测出系统的异常现象,并发出警报或给出处理对

策。FAGAN 等人开发了一个名为 VM 的集中医疗监视专家系统,可对装有呼吸装置的危重病人的生理学数据作解释,并自动对呼吸装置进行调节和控制。IBM 公司的 YES/MYS 专家系统用作计算机操作人员的助手,随时监视 MVS 这个复杂的操作系统的状态,自动检测并修复可能出现的软件故障。NILSON 等开发的 REACTOR 专家系统用来监测核反应堆的工作状态,有效地减轻了工作人员的负担,大大减小了事故和灾难发生的可能性。

#### 4. 专家系统在控制方面的应用

应用于控制的专家系统在实时动态环境下运行,它能对外部事件在给定的时间区间内迅速作出响应,自适应地对生产过程或被控对象进行监测和控制,以获得控制系统性能的改善和提高。进入 80 年代以来,专家系统技术在工业控制中的应用日益增多,各种类型的实时控制专家系统不断涌现。1983 年 Saridis 把智能控制用于机器人系统;1984 年 Dickey 等研制成功空间环境控制专家系统;1986 年 Wright 等推出了在微机上实现的用于卫星控制的混合专家系统控制器,1987 年美国 Foxboro 公司公布了新一代的智能分布式计算机自动控制系统;英国完成了 RESCU(实时专家系统用户协会)工程,并把其研究成果用于英国东北一个洗涤工厂的生产控制等方面。这些控制专家系统的成功应用显示了人工智能技术与控制工程相结合的巨大优越性。

#### 5. 专家系统在规划方面的应用

规划是根据预定的目标,制定一系列行动计划,如设计规划、生产调度、机器人动作规划等。美国 DEC 公司的 XCON 专家系统可用于规划计算机产品的配置,即如何根据用户的要求组合各种不同类型的计算机零部件来装配成一台完整的计算机。它接收用户输入的订货单,检验定货单的有效性,并做出相应的计算机配置计划。费根鲍姆的学生在听完这个故事后,兴趣很大,开发了几个小型的配置专家系统,用于许多不同的任务,如 PC 局域网的配置专家系统,帮助推销员推销局域网。后来这两个学生组建了 TRILOGY 公司,并将他们的产品扩充成一个叫做 Sales Builder 的系统,来帮助推销员推销许多种产品。该系统在美国获得了巨大成功。90 年代初期,伊拉克人侵科威特,开始了沙漠风暴之战,美国面临的一个重要问题就是如何尽快从美国和欧洲把 50 万军队 1500 磅的物资装备运送到沙特阿拉伯,为了解决这个由人工实施非常棘手的规划问题,美国军方使用早已着手开发的调度规划专家系统,很快完成了这些运输任务。美国航空航天局资助开发的宇宙飞船规划专家系统,使整个系统的发射活动周期缩短了百分之二十。

#### 6. 专家系统在银行业务决策中应用

信用卡在美国用得非常普通,但存在恶性透支和欺骗行为。由于这两个问题使美国每年蒙受 5 亿美元的损失,因此,迫切需要一个自动系统来决策是否允许顾客使用它的信用卡。当顾客用信用卡在商店买东西时,将信用卡放到磁阅读机中,卡中的有关信息经网络送到信息处理中心。那里存有关于信用卡使用的所有信息及历史记录。一般要求在 90 秒内做出是否允许顾客这次使用的决定。而这些记录在 PC 机上给出的是 16 屏幕的信息。要求人在 90 秒时间内,根据 16 屏信息做出决定几乎是不可能的。于是,一个名为 American Express 的银行公司开发了一个决策专家系统 AA 来解决这个问题。AA 可将信息量由 16 屏减少到 2 屏。第一屏信息中给出 AA 所做的决定,即是否允许使用,接下来是对该决定的解释;第二屏信息是支持所作决定的有关数据。有了 AA 的帮助,人们可以很快做出决定。AA 系统在 80 年代中期投入使用,有效地减少了恶性透支和欺骗行为,每年为 American Express 公司减少了几千万美元的损失。

## 第二节 专家系统的基本结构及工作原理

专家系统是以专家经验性知识为基础建立的,以知识库和推理机为中心的智能软件系统结构可表示为:

$$\text{知识} + \text{推理} = \text{系统}$$

经过二十多年的发展,专家系统的理论不断丰富和完善,出现了多种结构形式,最基本的结构还是以产生式系统为核心的结构,如图 11-1 所示,下面介绍各主要组成部分的功能:

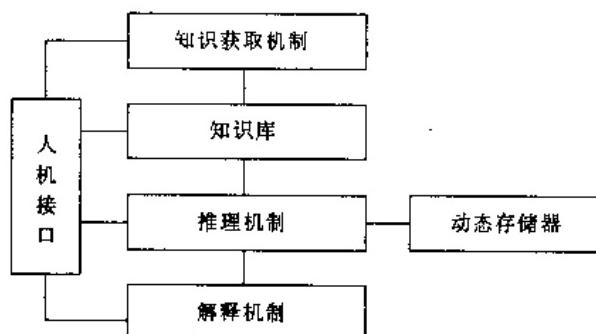


图 11-1 专家系统基本体系结构

### 1. 知识库

用于存储大量以产生式规则形式表示的领域专家的经验和知识以及已知的事实,规则形如:前件→后件

或 IF 条件 1 AND 条件 2……AND 条件 N THEN 动作或结论

例如,某计算机故障诊断专家系统的知识库中存储了数百条关于计算机故障诊断的产生式规则,其中的一条规则为:

RULE1: IF      外部电源插座电压正常  
          AND      计算机内电源输入电压为零  
          AND      电源插座电压正常  
          AND      电源插座到计算机的电源线完好  
          THEN      计算机的电源开关故障

为了表达专家知识的复杂概念,知识库中的规则往往分级存储,整个知识库形成一个树形结构,其中的规则也可嵌套,例如,在某动物识别专家系统中有如下三条规则形成了一个嵌套结构:

RULE1:IF      动物是哺乳动物  
          AND      动物是食肉动物  
          AND      动物是黄褐色  
          AND      动物身上有黑条纹  
          THEN      该动物是老虎  
RULE2:IF      动物有奶  
          THEN      该动物是哺乳动物  
RULE3:IF      动物吃肉

THEN 该动物是食肉动物

## 2. 动态存储器

动态存储器又称为工作存储器或者“黑板”，用于存放有关欲求解特定问题的已知事实、用户回答的事实、推理得到的中间结论等信息。在系统运行过程中，其存储内容是动态变化的。系统启动时，动态存储器是空的，不包含任何知识，随着推理的进行，系统不断把求解问题的初始状态及新的知识（即已知事实、用户回答的事实、推理中间结论等）存入动态存储器，这些新知识被用来测试和触发其他规则。随着每条规则的测试和触发，其新的结论和已知事实一起被添加到了动态存储器，供进一步推理时使用。同时，动态存储器还保存一次推理过程中的全部推理路径，供解释推理过程时使用。

## 3. 推理机制

推理机制主要有两个任务，一是推理，也叫知识的运用，即从知识库中已有的知识中推导出所需要的结论和知识；二是控制搜索过程，即确定知识库中规则的扫描顺序，决定在每个控制信息下要触发的规则，这又称为知识的选择。

本书前面的章节已经指出，有多种推理方法，以产生式系统为基础的专家系统采用的最基本的推理策略是反向推理（向后链接），用四个阶段的循环（产生式系统的循环周期）来控制规则的扫描，这四个阶段是：

（1）匹配（合一）。由顶向下（即从知识库中第一条规则开始）依次扫描知识库中所有规则，与动态存储器中的当前事实相对照，以搜索满足条件的规则，确定所匹配的目标；

（2）冲突消解（选择）。在发生冲突，即数条规则同时被匹配时，根据预先确定的评价准则，求出所冲突规则的优先度，决定触发（选择）哪一条规则；

（3）激活规则。调用匹配所触发规则的所有子目标的事实；

（4）动作。把所触发规则的结论添加到动态存储器。

运行时，推理机重复这四个阶段的循环，根据知识库中的知识及用户提供（回答）的事实，不断由已知的前提推出未知的结论记录到动态存储器，作为新的前提或事实继续推理过程，直到推出最终结论。

## 4. 解释机制

解释功能是专家系统区别于其他软件系统的重要特征之一。通过解释，可提高用户对专家系统的信赖程度，有助于专家系统的推广使用，也便于在使用中间发现专家系统的错误和漏洞，有助于测试、更新、维护专家系统。解释机制实现解释功能，在推理过程中回答用户关于系统的一些问题，如系统正在做什么，如何得到结论，为什么要作出某个决策，发出某个询问等，显示推理路径、解释推理过程。通过解释，可增强用户对系统的信任感，并起到教学的作用。有多种解释方法，常见的有：

（1）WHAT 解释方法，用于回答用户有关系统正在做什么的疑问。其实质是把事先考虑到的可能有疑虑的问题的解答存入专家系统程序内，需要时随时调出显示。

（2）HOW 解释方法，用于回答用户有关如何得到最终结论的问题。其实现策略是记录求解问题时的各个中间状态，必要时显示这些记录的内容。显示内容包括问题求解过程的推理路径，知识库中知识的使用情况等。

（3）WHY 解释方法，用于回答用户有关为什么做出某种决策或进行某个询问的问题。其实现策略是由底向上搜索目标树，直到找出问题的起源。

## 5. 知识获取机制

知识获取是建立知识库的重要基础,是专家系统开发中最关键也最艰难的一步,被称为专家系统开发的“瓶颈”。知识获取过程消耗人力、财力最多,约占专家系统开发总工作量的四分之一左右。许多著名的人工智能专家相继指出,专家系统的下一步是开发更好的知识获取工具。当前,知识获取有三种主要形式:

(1)人工获取。由知识工程师与领域专家会谈或查阅大量文献资料,收集、分析、归纳、整理领域知识。

(2)利用机器学习技术,从训练实例自动提取知识。

(3)使用文字识别和自然语言理解技术,自动阅读大量文献资料,自动提取知识。

在知识获取自动化方面,许多AI工作者进行了不懈的努力,取得了大量阶段性成果,但仍没有突破性的进展,当前大部分系统仍以手工知识获取为主。

知识获取机制的功能是辅助知识工程师和专家进行知识获取,即通过一些软件工具(如编辑器、学习程序等)把已有的知识(经验、事实、规则等),总结和提取出来,并自动转换为专家系统所能接受的内部表达形式。同时它还具有知识库维护功能,允许编辑修改知识库,对规则进行添加、删除、修改等操作。

## 6. 人机接口

用于控制人机交互过程,使用户能够以方便,直观的形式进行人机对话,同时充分发挥用户人机对话中的主观能动性,尽可能地避免用户的误操作。它具有输入输出两大功能,可把用户通过输入装置(如键盘、鼠标、扫描仪等)输入的信息经过变换送给系统,也可把系统的提示信息、中间及最终的推理结果、解释等信息转换成用户能够接受的形式送到屏幕或其他输出装置。有多种人机接口形式,如简单对话、菜单、窗口、图像符号、命令语言、自然语言等。简单对话是最简单的一种交互方式,系统就某一问题提出询问,要求用户以 yes 或 no 作为回答。这种方式容易使用,但操作速度过慢。为了方便使用,提高交互速度,在专家系统中也结合使用呈多级树形结构的下拉菜单接口,引导用户输入目标,减小搜索范围。并以图形、声音等直观、生动的形式进行人机交互。比较理想的人机接口是自然语言,已有人宣布自然语言是最终的人机对话类型。尽管在自然语言接口的实现上还有许多困难,一些具有受限的自然语言接口的专家系统已经出现。

## 第三节 专家系统的开发过程

专家系统是一个复杂的智能软件,与一般软件类似,但又有不同的特点。一般软件处理的对象是数值、文字、图形等信息,且有固定的算法序列,而专家系统软件处理的对象是以符号表示的知识,在运行过程中常有回溯发生,因此专家系统的开发过程与一般软件的开发有所不同。专家系统的创始人费根鲍姆教授把开发专家系统的技术称之为知识工程,即以知识获取、知识表示、知识运用(推理)为中心。根据这个思想,可把专家系统的开发过程分为以下几个阶段。

### 1. 问题定义与系统分析

在着手开发一个专家系统时,首先应了解清楚用户的需求,确定欲解决的问题的范围(领域),系统所要达到的目标,系统功能,性能指标,输入输出,使用对象,人机接口形式,运行软硬件环境等。在此基础上进行系统可行性论证及经济效益或社会效益分析。最后写出系统工作的数据(信息)流程图,写出分析报告及有关文档。

## 2. 知识获取

知识获取阶段的主要工作是根据所确定的系统目标及限定的问题求解范围,收集专家知识。在当前的技术条件下主要是通过走访多个领域专家及现场技术人员,查阅国内外大量文献资料来收集、归纳、整理领域知识。

## 3. 知识表示

知识表示阶段的任务是根据领域知识的性质,选择一种或组合数种知识表示方法,如前面介绍过的谓词逻辑表示法,框架表示法,产生式系统表示法等,把获取到的专家知识逻辑地表达出来,并以适当的形式存储到计算机中。

## 4. 软件实现

这一阶段的工作与一般软件设计类似,因此应遵循软件工程的原则,进行系统设计、编码、测试,建立知识库,实现推理机,动态存储器,人机接口,知识获取等专家系统程序模块,构成一个可运行的专家系统原型软件。

## 5. 系统测试与评价

费根鲍姆教授指出,专家系统开发是一个长期反馈的过程,对所生成的专家系统原型软件必须反复进行测试,评价,发现并改正其中的错误,才能使之实用。到目前为止,对专家系统的评价尚无统一标准,一些学者认为,测试与评价的主要内容包括:知识表示模式的选取是否恰当;知识库中的知识的正确性,完整性,一致性如何,知识库维护是否容易;所采用的推理方法和技术是否正确,推导出的结论是否可信,用户是否满意;人机接口使用是否方便,实用;系统的成本与效益情况等。

有多种系统评价方法,比较著名的是美国学者 Saaty 在 70 年代提出并广泛应用的层次分析评价法(AHP 法),它采用相对重要性加权,对系统进行多层次多指标的综合评价,具体步骤简述如下。

- (1)与领域专家具体协商确定系统目标及系统评价的指标体系;
- (2)把系统目标(如总体性能等)分解成层次结构;
- (3)由专家根据结构中每层各项元素的相对重要性给出其加权系数;
- (4)由顶向下逐层计算所有元素相对于总目标的权重,据被比较方案的总权重,作出最终判决决策。

当然,实践是检验系统性能的最根本的标准,系统测试与评价最终应到专家系统运行的现场(真实环境)中去进行,接受实践的检验。

# 第四节 PC 计算机故障诊断指导专家系统(PCDGES)示例

PCDGES 是作者在 80 年代中期开发的一个基于产生式的咨询专家系统,用 TURBO PROLOG 语言编码,可在 PC 系列计算机,CCDOS 环境下运行,下面简要介绍其开发过程。

## 1. 系统目标及工作范围

PCDGES 的目标是为计算机初级用户提供一个有关 PC 计算机故障的咨询工具软件,回答用户有关 PC 计算机故障的问题,并提出相应的故障处理对策。系统的工作(诊断)范围仅限于部件级的故障诊断咨询。

## 2. 知识获取

PCDGES 的知识获取通过手工方式进行。通过总结作者的经验并查阅大量文献资料,得

到了有关 PC 计算机故障的大量经验性知识,分层画出了 PC 计算机系统故障诊断流程图,包括:主诊断、检查测试, RAM 检查, 键盘检查, 主板检查, 硬盘检查, 软盘检查, 启动检查, 叫声检查, 电源检查, 机房条件检查等 15 个故障诊断流程图。其中的主诊断及检查测试诊断流程示意图分别如图 11-2, 图 11-3 所示。在此基础上, 收集整理并列出了故障现象与可能的故障部位对照表, 其中的一些项目列举如下:

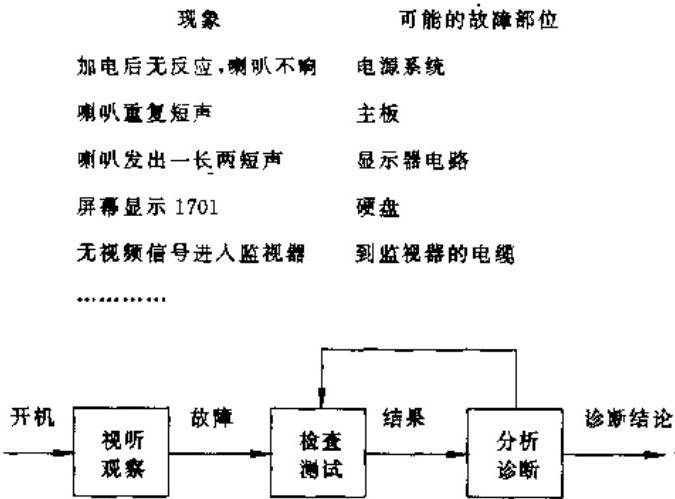


图 11-2 计算机故障主诊断流程示意图

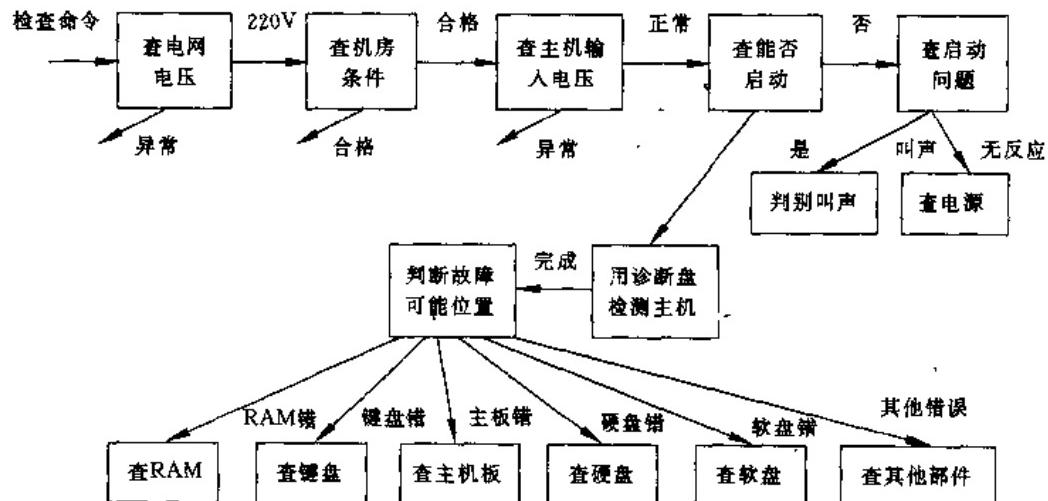


图 11-3 故障检查测试流程示意图

### 3. 知识表示

根据问题及所获取知识的特点, 本系统采用产生式规则表达诊断知识, 整个系统的知识分级分块存放, 所建立知识库结构呈非对称树形结构, 如图 11-4 所示。图中的空白方框表示存储特定硬件模块诊断知识的分级规则库, 小圆圈表示最底层的诊断规则, 即知识库的叶子结点。

知识库中一些典型规则如下:

RULE 1: IF 第 1 类系统错,  
 AND 第 1 类启动错,  
 AND 第 4 类报错叫声,

THEN 系统板有问题,应修理或更换  
 RULE 2: IF 启动不正常  
     THEN 第 1 类系统错  
 RULE 3: IF 启动时出现加电检查错(POST 失败)  
     THEN 第 1 类启动错  
 RULE 4: IF 一声长叫,--声短叫  
     THEN 第 4 类叫声  
 .....

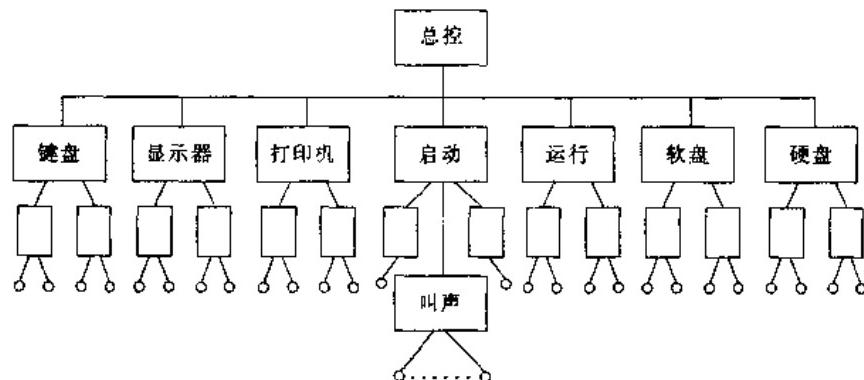


图 11-4 PCDGES 知识库结构简图

#### 4. 软件实现

在知识表示模式确定后,也就确定了系统结构的基本形式。PCDGES 采用的知识表示模式是产生式,所以整个系统以产生式系统为基础,其总体结构与图 11-1 大体类似。系统使用 TURBO PROLOG 语言,在 PC 386 计算机上编码实现。下面介绍所实现 PCDGES 软件的主要模块。

##### (1) 知识库

知识库存放了 300 多条规则及大量事实,每条规则及事实被编码为数条 PROLOG 谓词子句。例如,本节 3 所举 4 条产生式规则,被编码为以下 5 条 PROLOG 规则:

```

RULE 1:diagm("系统板");- (结论)
 checkc("系统",'1'), (检查是否第 1 类系统错)
 checkc("启动",'1'), (检查是否第 1 类启动错)
 checkc("叫声",'4'), (检查是否第 4 类报错叫声)
 cause("系统板"). (显示结论信息)

RULE 2:checkc("系统",'1');-
 cond(1,"系统启动失败").

RULE 3:checkc("启动",'1');-
 cond(2,"加电后屏幕无反应"),
 cond(3,"机内电源电压正常").

RULE 4:checkc("叫声",'4');-
 cond(4,"系统启动失败"),

```

cond(5,“喇叭一声长叫,一声短叫”).

RULE 5:cause(“系统板”):-

write(“系统板有问题,应修理或更换”).

## (2) 推理机(规则解释器)

由于PC机的大多数故障现象及诊断推理逻辑是确定性的,如某个信号的有无,磁盘马达是否转动等,很少有模糊不清的事实,为了简化系统,减少系统开销,本系统采用了反向为主正向为辅的精确推理策略。推理机运行时先反向推理,即假设一个可能成立的子目标(故障结论),试图用知识库或动态数据库的事实证明它为真;若证明失败,则系统向前搜索(正向推理),直到发现另一个假设(故障结论),再重复使用反向推理,试图证明它为真。这个过程周而复始,直到所有目标被证明为真或所有可能的路径都已测试。

## (3) 解释机制

PCDGES解释机制采用的策略类似于第二节所述的WHY解释方法,其实现的基本程序框图如图11-5所示。在该模块的实现过程中,应用了TURBO PROLOG的模式匹配功能,字符串处理功能,窗口管理功能、类型转换功能等。

## (4) 人机接口

PCDGES的人机接口采用交互式人机接口,系统运行时的所有人机交互过程均在分级彩色汉字菜单及中文提示信息下进行。系统菜单分为四级,一级为系统总控菜单;二级为诊断主控菜单;三级为诊断范围选择及知识库选择菜单;四级是故障现象选择,咨询对话等菜单。整个菜单系统呈树形结构,如图11-6所示,其中四个典型的菜单如图11-7所示。

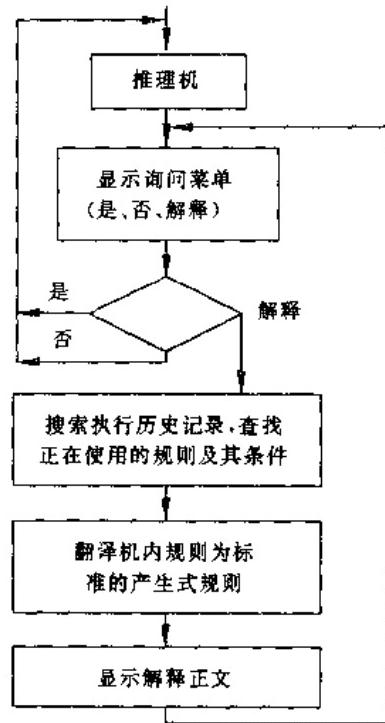


图 11-5 PCDGES 解释机制基本  
程序框图

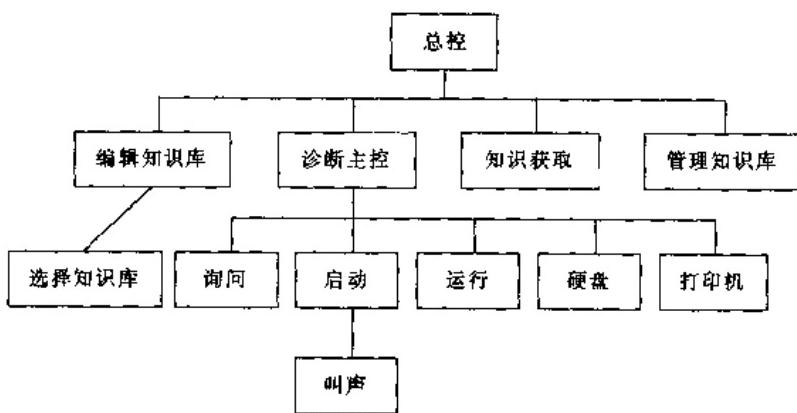


图 11-6 PCDGES 人机接口基本结构图

PCDGES系统启动后,屏幕首先显示系统总控菜单,用户从总控菜单开始逐级进行选择,直到选定了叶子结点菜单的一项后,就给了PCDGES一个执行目标,从而驱动推理机求解此目标。求解过程中,系统询问用户一系列问题,要求用户选择询问菜单中(仅有是,否,解释三项内容)的一项作为回答。若用户明白系统的提问,可以是或否回答,否则用户可选择解释项,屏

幕即显示出关于此询问的解释信息,然后重复显示问题及询问菜单,再次要求用户回答,.....直到求解目标完成,屏幕再次显示系统总控菜单,等待下一次人机对话过程的开始。这样的树形菜单结构不仅使用方便,而且由于通过用户逐级选择限定了搜索范围,减少了搜索过程中的冗余目标状态,大大提高了推理速度。

#### (5)知识获取机制

PCDGES 的知识获取机制的基本结构如图 11-8 所示。该机制向用户提供了一个类似 WORDSTAR 的全屏幕编辑器,可用于编辑、修改整个知识库;同时还有一个独立于推理机的半自动化知识获取智能接口,可以“告知”学习的方式进行知识获取。用户通过与 PCDGES 系统对话,“告知”系统必要的知识(信息),智能接口就把这些知识自动转化为专家系统的内部表示形式而存入知识库。

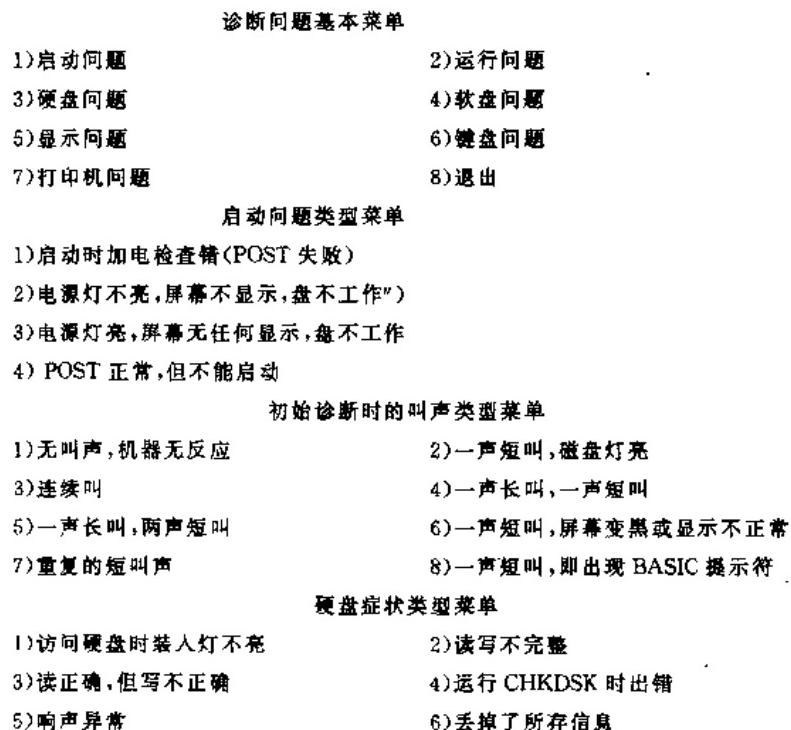


图 11-7 PCDGES 的四个典型菜单

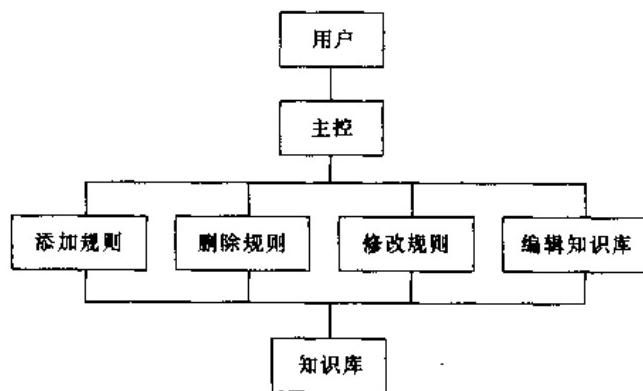


图 11-8 PCDGES 知识获取机制的基本结构图

## 习题十一

1. 简述专家系统概念、功能；画出基于产生式的专家系统基本结构框图。
2. 简述基于产生式的专家系统推理循环周期，指出制约其推理速度的关键环节。
3. 写出开发专家系统的基本步骤，指出影响专家系统实用性的主要因素。如何测试与评价一个专家系统？
4. 某人欲开发一个专家系统，但收集到的规则太少，而有大量历史数据，试问用何种结构建造此专家系统最好？
5. 为了从以下方面改进专家系统的性能，应分别采取何种措施：
  - (1) 开发速度；
  - (2) 运行速度；
  - (3) 实时性能；
  - (4) 学习能力；
  - (5) 人机交互。
6. 在常规 CAD/CAM 系统及控制系统中嵌入专家系统有什么好处？

## 第十二章 自然语言处理

自然语言处理是人工智能领域中早期较活跃的研究领域之一。由于它的难度很大,至今仍未能达到很高的水平。

本章首先介绍自然语言处理的概念和基本理论,然后从自然语言理解和自然语言生成两个方面分别讨论各种处理方法,最后给出自然语言处理系统的设计思想和设计过程。

### 第一节 自然语言处理的一般问题

什么是自然语言处理?自然语言处理是如何发展的?自然语言处理研究中有哪些学术观点?等等。这些问题是在开始研究自然语言处理时应当首先了解的。

#### 一、自然语言处理的概念及意义

自然语言指人类语言集团的本族语,如汉语、英语、日语等。众所周知,语言是思维的载体,人类历史上以语言文字形式记载和流传的知识占到知识总量的 80%以上。就计算机应用而言,有 85%左右都用于语言文字的信息处理。在信息化社会中,语言信息处理的技术水平和每年所处理的信息总量已成为衡量一个国家现代化水平的重要标志之一。

自然语言处理,一方面,可以定义为计算机处理人类在日常生活中使用的自然语言——书面或口头的能力,另一方面,可定义为认知科学中研究人类语言行为的一个分支。

自然语言处理作为语言信息处理技术的一个高层次的重要研究方向,一直是人工智能领域的核心课题之一。如果计算机能够理解、处理自然语言,人-机之间的信息交流能够以人们所熟悉的本族语言来进行,将是计算机技术的一项重大突破。另一方面,由于创造和使用自然语言是人类高度智能的表现,因此对自然语言处理的研究也有助于揭开人类高度智能的奥秘,深化对语言能力和思维本质的认识。自然语言处理这个研究方向在应用和理论两方面都有重大意义。

#### 二、自然语言处理的发展简史

60 年代以来已经产生过一些成功的自然语言理解系统,用来处理受限的自然语言子集。这种子语言或是在句子结构的复杂性方面受到限制(句法受限),或是在所表达的事物的数量方面受限(语义受限,或领域受限)。其中有一些系统,如人机接口和机器翻译系统,已成为市场上的商品。但要想让机器能像人类那样自如地运用自然语言,仍是一项长远而艰巨的任务。

30 年来自然语言理解的研究大体上经历了三个时期:即 60 年代以关键词匹配为主流的早期,70 年代以句法-语义分析为主流的中期,和 80 年代开始的基于知识的新一代自然语言处理系统。目前,新提出的基于大规模语料库的自然语言处理思想正在蓬勃发展。

### 1. 以关键词匹配为主流的早期历史

60 年代开发的自然语言理解系统,大都没有真正意义上的语法分析,而主要依靠关键词匹配技术来识别输入的句子的意义。在有些系统中事先存放了大量包含某些关键词的模式,每个模式都与一个或多个解释(又叫响应式)相对应。系统将当前输入句子同这些模式逐个匹配,一旦匹配成功便立即得到了这个句子的解释,而不再考虑句子中那些不属于关键词的成分对句子意义会有什么影响。所以这是一种近似匹配技术,它的最大优点是允许输入的句子不一定要遵循规范的语法。但这种分析技术的不精确性也正是这种方法的主要弱点,往往会导致错误的分析。

这一时期的几个著名系统是:1968 年 B. Raphael 在美国麻省理工学院完成的 SIR(Semantic Information Retrieval,语义信息查询)系统,它能记住用户通过英语告诉它的事实,然后对这些事实进行演绎,回答用户提出的问题。同一年,J. Weizenbaum 在美国麻省理工学院设计的 ELIZA 系统,能模拟一位心理治疗医生(机器)同一位患者(用户)的谈话。

### 2. 以句法—语义分析为主的中期历史

采用这种思想的自然语言处理系统的典型系统框图如图 12-1 所示。

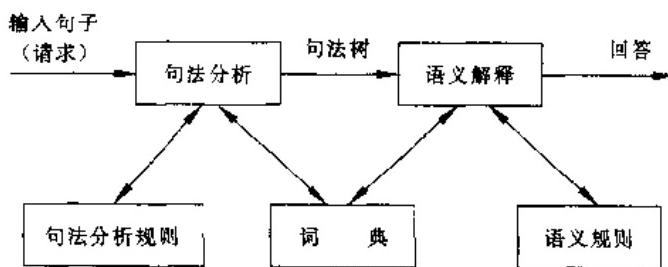


图 12-1 基于句法—语义分析的系统框图

这个时期的代表系统 LUNAR 是第一个允许用普通英语同计算机数据库对话的人机接口,是 1972 年美国 BBN 公司的 W. Woods 负责设计的,用于协助地质学家查找、比较和评价阿波罗 11 飞船带回的月球标本的化学分析数据。同年,T. Winograd 设计了 SHEDLU 系统,是一个在“积木世界”中进行英语对话的自然语言理解系统。它把句法、推理、上下文和背景知识灵活地结合于一体,模拟一个能够操纵桌子上一些玩具积木的机器人手臂,用户通过人—机对话方式命令机器人放置那些积木块,系统通过屏幕给出回答并显示现场的相应情景。

### 3. 基于知识的语言处理系统

这一时期的主要特征是引入了知识的表示和处理方法,引入了领域知识和推理机制,借鉴了许多人工智能和专家系统中的思想,使自然语言处理系统不再局限于单纯的语法规则的研究,而与它所表示的客观世界紧密结合在一起,极大地提高了系统处理的正确性。

我们知道,语言是人类智能、思想的直接外在表现,与所有的智能活动一样,是以知识为基础的。语言与客观世界的关系如图 12-2 所示,图中上方的第一个三角形就是著名的语义三角形,左下方的三角形构成现代语言学中另一个三角关系。“知识(语义)”在这两个三角形中起着举足轻重的地位。通过这两个三角关系,可以清楚地看到语言文字是如何反映客观世界的,看到知识的核心地位。

进入 80 年代以来,自然语言处理系统越来越趋向于实用化与工程化,出现了一批商品化的自然语言人—机接口和机器翻译系统。著名的有美国人工智能公司(AIC)生产的英语人—机接口系统 Intellect,美国弗雷公司生产的 Themis 人—机接口;欧洲共同体在美国乔治敦大学

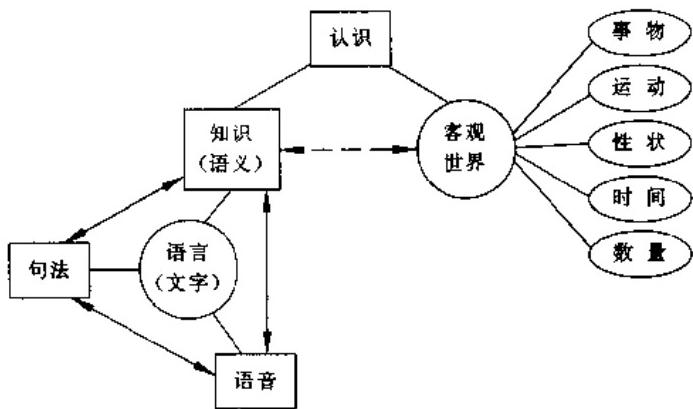


图 12-2 语言与概念

开发的机译系统 SYSTRAN 的基础上成功地实现了英、法、德、西、意、葡等多语对的机器翻译系统，等等。

#### 4. 基于大规模语料库的自然语言处理系统：

几十年的自然语言处理研究实践证明，在当今计算技术条件下，要想把处理自然语言所需的知识（包括语言学知识和语言学以外的世界知识）都用现有的知识表示技术明确表达出来，是不可能的。这既是由于这种知识的“数量”巨大，又是由于它们在“质”的方面高度的不确定性和模糊性。

为了处理大规模的真实文本，最近十年来新提出了语料库语言学（Corpus Linguistics）。它顺应了大规模真实文本处理的需求，提出了以计算机语料库为基础的语言学研究及自然语言处理的新思想。它认为语言学知识的真正源泉是大规模的来自于生活的语料，计算语言学工作者的任务是使计算机能够自动或半自动地从大规模语料库中获取处理自然语言所需的各种知识，他们必需客观地而不是主观地对库存的语言事实作出描述。

80 年代英国兰开斯特大学 Leech 领导的 UCREL 研究小组，利用已带有词类标记的布郎语料库，经过统计分析得出一个反映任意两个相邻标记出现频率的“概率转移矩阵”。他们设计的 CLAWS 系统依据这种统计信息，而不是系统内存储的知识，对 LOB 语料库约一千万词的语料进行词类的自动标注，成功率高达 96%。CLAWS 系统的成功使许多研究人员相信，基于语料库的处理思想能够在工程上、在宽广的语言覆盖面上解决大规模真实文本处理这一极其艰巨的课题，至少也是对传统的处理方法一个强有力的补充。

### 三、自然语言处理领域中的几种思想

从不同的角度，以不同的观点来看待自然语言，必然导致不同的研究方法的研究思路。下面提出的几种观点，反映当前对于自然语言的科学理解，给出了这一研究领域中的基本处理思想，如图 12-3 所示。

认知学观点来源于把人类视为一种高级信息处理系统，强调对于人类智能活动的研究以及在计算机上的模拟和实现。语用学观点把语言视为人与人之间的通信媒介，任何对于话语的理解或生成都不能脱离该话语存在的前后语境和该话语使用者的心理背景。语言学观点是对于自然语言进行研究的最初也是最基本的方法，强调对于句子结构和语法的研究，重点在于描述语言，寻找或构造一组能够包含尽可能多的语言现象的普遍适用的语法规则。

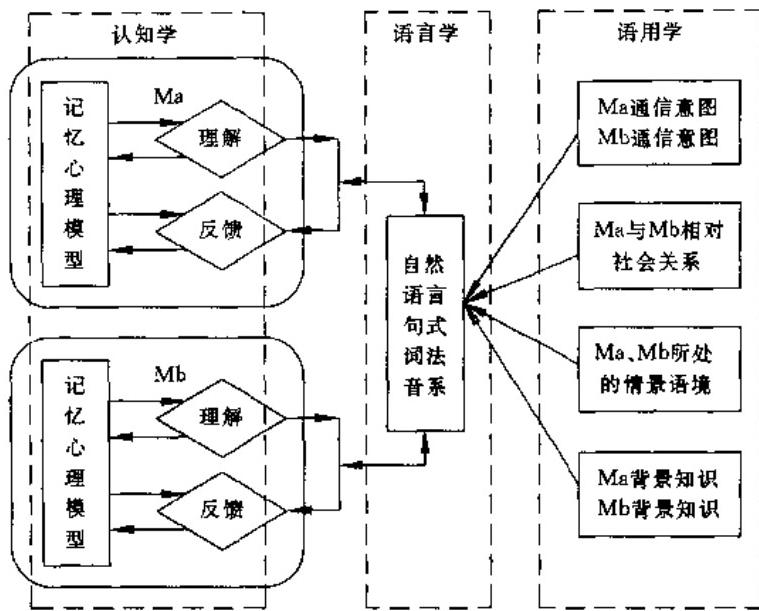


图 12-3 自然语言处理的不同研究角度

这三种角度反映了语言的三个基本方面：语言概念的最初选择和意义的最终还原所涉及到的人类意识或潜意识中的处理机制；自然语言的功能，是服务于人际交往；语言的使用必须遵守一定的规范，有着被特定的语言集团接受的表层形式。观点不同，也就决定了具体语言处理系统中所采取的技术思想和实现方法的不同。

#### 四、自然语言处理的层次

语言虽然表示成一连串文字符号或一串声音流，但其内部事实上是一个层次化的结构，从语言的构成中就可以清楚的看到这种层次性。一个文字表达的句子是由词素→词或词形→词组或句子构成，而声音表达的句子则是由音素→音节→音词→音句构成，其中每个层次都是受到语法规则的制约。因此，语言的处理过程也应当是一个层次化的过程。许多现代语言学家把这一过程分为五个层次：语音分析、词法分析、句法分析、语义分析和语用分析。虽然这种层次之间并非是完全隔离的，但这种层次化的划分的确有助于更好地体现语言本身的构成，并且在一定程度上使得自然语言处理系统的模块化成为可能。

##### 1. 语音分析

在有声语言中，最小可独立的声音单元是音素，音素是一个或一组音，它可与其他音素相区别。如 pin 和 bin 中分别有/p/和/b/这两个不同的音素，但 pin、spin 和 tip 中的/p/是同一个音素，它对应了一组略有差异的音。语音分析则是根据音位规则，从语音流中区分出一个个独立的音素，再根据音位形态规则找出一个个音节及其对应的词素或词。

##### 2. 词法分析

词法分析的主要目的是找出词汇的各个词素，从中获得语言学信息，如 unchangeable 是由 un-change-able 构成的。在英语等语言中，找出句子的一个个词汇是一件很容易的事情，因为词与词之间是有空格来分隔的。但要找出各个词素就复杂的多。而在汉语中的每个字就是一个词素，所以要找出各个词素是相当容易的，但要切分出各个词就非常困难。如“我们研究所有东

西。”，可以是“我们一研究所一有东西。”，也可以是“我们一研究…所有一东西。”。

通过词法分析可以从词素中获得许多语言学信息。英语中构成词尾的词素“*s*”通常表示名词复数，或动词第三人称单数，“*ly*”是副词的后缀，而“*ed*”通常是动词的过去时过去分词等，这些信息对于句法分析是非常有用的。另一方面，一个词可有许多的派生、变形，如 *work*，可变化出 *works*, *worked*, *working*, *worker*, *workable* 等。这些词如果全放入词典将是非常庞大的，而它们的词根只有一个。

### 3. 句法分析

句法分析是对句子和短语的结构进行分析。句法分析的最大单位是一个句子。分析的目的就是找出词、短语等的相互关系以及各自在句中的作用等，并以一种层次结构来加以表达。这种层次结构可以反映从属关系、直接成分关系，也可以是语法功能关系。自动句法分析的方法很多，有短语结构文法、格语法、扩充转移网络、功能语法等。

### 4. 语义分析

理解语言的核心是理解语义。随着自然语言处理的发展，越来越多的研究者开始侧重于语义层的研究，句法则退居到第二位。

对于语言中的实词而言，每个词都是用来称呼事物，表达概念。句子是由词组成的，句子的意义与词义直接相关，但不等于词义的简单相加。“我打他”与“他打我”，词汇完全相同，但表达的意义完全相反。因此，还应考虑句子的结构意义。语义分析就是要找出词义、结构意义及其结合意义，从而确定语言所表达的真正含义或概念。在自然语言处理中，语义愈来愈成为一个重要的研究内容。

### 5. 语用分析

语用就是研究语言所存在的外界环境对语言使用所产生的影响。它描述语言的环境知识，语言与语言使用者在某个给定语言环境中的关系。关注语用信息的自然语言处理系统更侧重于讲话者/听话者模型的设定，而不是处理嵌入到给定话语中的结构信息。研究者们提出了很多语言环境的计算模型，描述讲话者和他的通信目的，听话者及他对说话者信息的重组方式。构建这些模型的难点在于如何把自然语言处理的不同方面以及各种不确定的生理、心理、社会、文化等背景因素集中到一个完整的连贯的模型中。

讨论自然语言的分层结构，讨论每层的信息处理任务和功能，及在整体结构中的位置。这对全面理解 NLP 领域中的各种处理方法会有很大帮助。在表 12-1 中，给出了自然语言的层次划分，其理论依据和常见的实现技术。

表 12-1 自然语言的层次划分及对应技术

| 理 论                        | 层 次 结 构 | 具 体 实 现 技 术                 |
|----------------------------|---------|-----------------------------|
| 模 板 匹 配<br>基 于 规 则         | 语 音     | 模 式 匹 配 ①                   |
| 基 于 词 素<br>基 于 词 汇         | 词 汇     | 词 典 构 造                     |
| 转 换 产 生 语 法<br>词 汇 功 能 语 法 | 语 法     | 扩 展 转 移 网 络 (ATN)<br>CF 规 则 |

① 目前的自然语言处理系统中很少包含语音处理模块，因为在传统观点中，语音识别属于信号处理和电子工程学领域，而不是语言学的研究范畴。

续表 12-1

| 理 论                       | 层 次 结 构 | 具 体 实 现 技 术           |
|---------------------------|---------|-----------------------|
| 格语法<br>语义基元理论<br>模型理论     | 语义      | 产生式规则<br>概念相依理论<br>剧本 |
| 基于记忆的推理<br>言语行为理论<br>篇章语法 |         | 框架<br>语义网络<br>逻辑      |
|                           | 语用      |                       |
|                           |         |                       |

本章将按照这种层次划分的顺序,依次介绍语法层、语义层和语用层的代表性理论,最后给出自然语言处理系统设计的总体思路。其中语音和词汇两个层次不作介绍,原因是语音处理一般认为属于电子工程学领域,不是语言学的研究重点;而词汇对具体语言有很强的依赖性,难以抽象介绍。

## 第二节 语法层: 形式语法分析

上节中介绍了语言的不同处理思想和语言的内在层次,本节将讨论在自然语言处理中长期占主导地位的形式语法问题。重点介绍形式语法理论的代表,乔姆斯基(Noam Chomsky)的转换生成语法及其典型的实现方式,扩充转移网络 ATN。

### 一、转换生成语法

乔姆斯基的转换生成语法实际由两部分组成:生成语法和转换语法。下面分别对它们进行介绍。

#### 1. 乔姆斯基的生成语法

乔姆斯基指出人类语言具有无限性,无论一个人已经听到或说过多少语句,总还有更多他尚未接触过的语句。当一个人学习一种语言时,并不是通过接触语言的所有语句,而是学习语言的内在结构知识。他创立了“生成语法”,又称为“短语结构语法”,以试图描述这种内在的语言结构。这是一种严格形式化的规则系统,对自然语言进行描述,无需任何未在系统中明确表示的附加信息,就能够生成符合所描述语言语法规范的自然语言语句,并且为每个句子赋予一个结构化描述,或称之为短语标记。

形式语言理论中的语法指一组能够描述或生成部分自然语言语句的规则集。短语结构语法中的语法或规则系统表示为一系列的重写规则,定义了由语法所限定的自然语言中各种构成元素间的关系,而“结构化描述”通常表示为一种树形结构。

下面给出了一个简单的短语结构语法:

S→NP+VP  
NP→Det+N  
VP→VP+ADV  
VP→Aux+V  
ADV→PP  
ADV→Adv

$PP \rightarrow Prep + NP$   
 $Det \rightarrow a, the, this, \dots$   
 $N \rightarrow dog, baby, park, \dots$   
 $Aux \rightarrow must, can, \dots$   
 $V \rightarrow run, smile, hurt, \dots$   
 $Adv \rightarrow quickly, slowly, \dots$   
 $Prep \rightarrow in, with, by, \dots$

该语法可生成如下的语句：

The dog can run in the park.

上述语法中给出了该句的短语标记，也就是它的句法结构如图 12-4 所示。

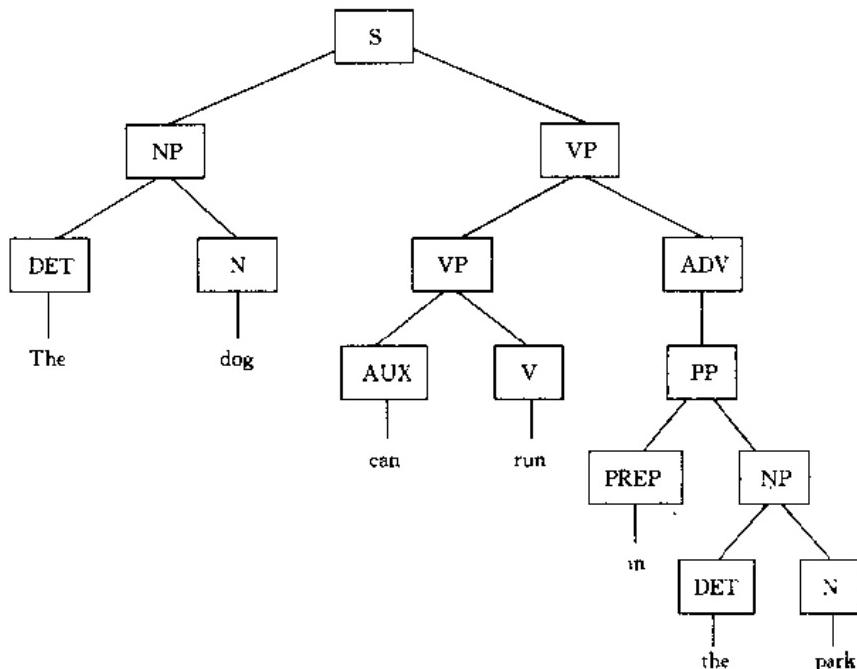


图 12-4 短语标记举例

## 2. 短语结构语法 $G$ 的形式化定义

短语结构语法  $G$  的形式化定义如下：

$$G = (T, N, S, P)$$

其中： $T$  是终结符的集合，终结符是指被定义的那个语言的词（或符号）；

$N$  是非终结符号的集合，这些符号不能出现在最终生成的句子中，是专门用来描述语法的。显然， $T$  和  $N$  的并集构成了符号集  $V$ ，而且  $T$  和  $N$  不相交，因此有： $V = T \cup N$

$$T \cap N = \emptyset$$

$S$  是起始符，它是集合  $N$  中的一个成员；

$P$  是一个产生式规则集。每条产生式规则具有如下的形式：

$$a \rightarrow b$$

其中  $a \in V^*$ ,  $b \in V^*$ , 且  $a \neq b$ ;  $V^*$  表示由  $V$  中的符号所构成的全部符号串（包括空符号串  $\emptyset$ ）的集合， $V^+$  表示  $V^*$  中除空符号串  $\emptyset$  之外的一切符号串的集合。

在一部短语结构语法中，基本运算就是把一个符号串重写为另一个符号串。如果  $a \rightarrow b$  是

一条产生式，就可以通过用  $b$  来置换  $a$ ，重写任何一个包含子串  $a$  的符号串，记作“ $\Rightarrow$ ”。所以，如果  $u, v \in V^*$ ，有  $uav \Rightarrow ubv$ ，就说  $uav$  直接产生  $ubv$ ，或  $ubv$  由  $uav$  直接推导得出。以不同的顺序使用产生式规则，就可以从同一符号产生许多不同的串。由一部短语结构语法定义的语言就是可以从起始符  $S$  推导出来的全部终结字符串的集合。

一般来说，如果把语法  $G$  所生成的语言记作  $L(G)$ ，则

$$L(G) = \{W \mid W \in T^*, \text{且 } S \Rightarrow W\}$$

这条定义的意思是，对于所有的符号串  $W$ ，如果  $W$  是由终结符所组成的串，且用语法  $G$  可以从起始符  $S$  中推导出  $W$ ，那么符号串  $W$  的集合就是由语法  $G$  所生成的语言  $L(G)$ 。

换言之，一个符号串要属于  $L(G)$  必须满足以下两个条件：

- (1) 该符号串只包含终结符；
- (2) 该符号串能根据语法  $G$  从起始符  $S$  推导出来。

### 3. 乔姆斯基定义的四种形式语法

根据形式语法中所使用的规则集的不同，乔姆斯基定义了四种类型的语法：

- (1) 无约束短语结构语法，又称 0 型语法；
- (2) 上下文有关语法，又称 1 型语法；
- (3) 上下文无关语法，又称 2 型语法；
- (4) 正则语法，又称 3 型语法。

型号愈高所受约束愈多，生成能力就愈弱，能生成的语言集就越小。下面简要讨论这几类语法。

3 型语法，又称正则语法，或称有限状态语法，只能生成非常简单的句子。

正则语法有两种形式：左线性语法和右线性语法。在一部左线性语法中，所有规则必须采用如下形式：

$$A \rightarrow Bt$$

或

$$A \rightarrow t$$

其中  $A, B \in N, t \in T$ ，即  $A, B$  都是单独的非终结符， $t$  是单独的终结符。而在一部右线性语法中，所有规则必须如下书写：

$$A \rightarrow tB$$

或

$$A \rightarrow t$$

对于一部正则语法，总能用有限状态图来表示，如图 12-5 所示。图上每个带标记的结点对应一个非终结符，另外一个特殊结点叫做最后结点，用带斜线的圆表示。每个结点对应于生成中的一个状态。对于每条形如  $A \rightarrow Bt$  的规则，便从结点  $A$  到结点  $B$  画一条弧，并在弧上标注终结符  $t$ 。对每条形如  $A \rightarrow t$  的规则，则从结点  $A$  到最后结点画一条弧，并在弧上标注终结符  $t$ 。

为了生成被一部正则语法所定义的语言中的一个句子，只需在与其对应的有限状态转移图上，从起始结点开始，沿着任何一条弧从当前结点转移到下一个新结点，并记下该弧上标注的符号。当到达最后结点时，记下的符号串便是这种语言的一个句子。换言之，在转移状态图上每一条从起始结点到最后结点的路径都对应于被这部语法所生成的语言中的一个句子。

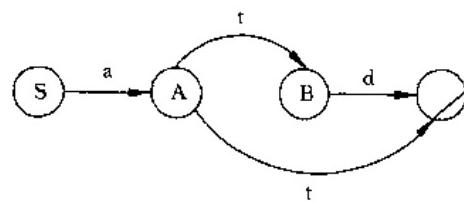


图 12-5 一个有限状态图

因此,这部语法又称为有限状态语法。“有限状态”指的是图中结点的数量是有限的。当我们处于一个句子的生成过程中,为了正确地结束这个句子,需要知道的唯一信息就是当前所处的状态(结点),而无需了解已经生成的那部分句子的其他任何句子。

正则语法之所以引入注目是因为有限状态网络提供了一种可以用来生成和分析语言的简单机制。

上下文无关语法的生成能力略强于正则文法。在一部上下文无关语法中,每一条规则都采用如下的形式: $A \rightarrow x$

其中  $A \in N, x \in V^*$  [ $V^* = (T \cup N)^*$ ],即每条产生式规则的左侧必须是一个单独的非终结符。在这种体系中,规则被应用时不依赖于符号  $A$  所处的上下文,因此称为上下文无关语法。

显然, $S \rightarrow (S)$  是上下文无关语法中一条合法的规则,可以生成正则语法所不能解决的嵌套结构。

上下文有关语法是一种满足以下约束的短语结构语法:对于每一条形式为

$$x \rightarrow y$$

的产生式, $y$  的长度(即符号串  $y$  中的符号个数)总是大于或等于  $x$  的长度,而且  $x, y \in V^*$  [ $V^* = (T \cup N)^*$ ]。因此,

$$AB \rightarrow CDE$$

是上下文有关语法中一条合法的产生式,但

$$ABC \rightarrow DE$$

不是。这一约束可以保证上下文有关语法是递归的,在读入一个符号串后能最终判断出这个字符串是或不是由这种语法所定义的语言中的一个句子。

上下文有关指对非终结符进行替换时,需要考虑该符号所处的上下文环境。对于产生式:

$$a_1 A a_2 \rightarrow a_1 y a_2 (A \in N, y \neq \emptyset, a_1, a_2 \text{ 不同时为 } \emptyset)$$

当用替换  $A$  时,只能在上下文为  $a_1$  和  $a_2$  时才可进行。

如果没有对于短语结构语法的产生式规则的两边做更多的限制,仅要求  $x$  中至少含有一个非终结符,即成为乔姆斯基体系中生成能力最强的一种形式语法,即无约束短语结构语法。

$$x \rightarrow y \quad (x \in V^+, y \in V^*)$$

0型语法是非递归的语法,即无法在读入一个符号串后最终判断出这个字符串是或不是由这种语法所定义的语言中的一个句子。因此,0型语法很少用于自然语言处理。

#### 1. 乔姆斯基的转换语法

上面介绍了乔姆斯基定义的各种类型的生成语法,下面介绍其转换理论。所谓“转换”(transformation)就是把具有共同成分的句子成对地联系起来,例如“主动句—被动句”、“陈述句—疑问句”、“两个简单句—复合句”等。这些相应的句子所带的信息相同,只是强调的重点或侧面不同。因此,在自然语言处理中,可以只处理若干核心句子,而使用一条或几条转换规则把它们转换为其他对应的句子。如把主动句转换为被动句,陈述句转换为疑问句等。

乔姆斯基的转换语法由一个基础部件和一个转换部件组成。基础部件是一部上下文无关语法,它产生一组表示句子深层结构的树,对应于句子所带有的信息。转换部件是一组重写规则,把它们应用于一棵深层结构树,便得到一棵或多棵表层结构树。表层结构树的前沿就是语言的句子。

每条转换规则主要由结构索引(structural index)和结构变换(structural change)两部分组

成。如“被动”转换规则：

规则中首先列出可应用“被动”转换规则的句子必须具有的成分：

$[NP, Aux, Vt, NP, (Adv)]$

其结构索引为： $x_1 \ x_2 \ x_3 \ x_4 \ x_5$

其结构变换为： $x_1 \ x_2 \ x_3 \ x_4 \ x_5 \rightarrow x_4 \ x_2 \ x_1 \ be + en + x_3 \ by + x_1 \ x_5$

如句子：  
*The car will hit that tree soon.*

$NP \ Aux \ Vt \ NP \ (Adv)$

可以应用上述规则，变换为：

*That tree will be + en + hit by the car soon.*

## 二、扩充转移网络

70年代由伍兹(W. Woods)提出来的，曾应用于他著名的LUNAR系统中，后来，卡普兰(Kaplan)对其做了一些改进。ATN语法属于一种增强型的上下文无关语法，概念来源于有限状态转移网络，它在自然语言理解中有着广泛的应用。

ATN是由一组网络所构成的，每个网络都有一个网络名，每条弧上注明扩展条件和相应操作。ATN弧上的标记也可以是其他网络的标记名，因此ATN网络是一种递归网络。在ATN网络还有一种空弧jump，它不对应一个句法成分也不对应于一个输入词汇。

ATN网络的条件和操作采用寄存器的方法来实现。ATN的每个寄存器由两部分构成：句法特征寄存器和句法功能寄存器。在特征寄存器中，每一维特征都有一个特征名和一组特征值，以及一个缺省值来表示。如“数”的特征维可以由两个特征值“单数”和“复数”，缺省值可以是空值。功能寄存器则反映了句法成分之间的关系和功能。

自然语言语句的理解过程就是程序生成其对应分析树的过程。在分析树的各个节点上都放上寄存器，用来存放该节点的句法功能和句法特征，程序利用ATN网络中给出的条件和操作不断地对它们进行访问和设置。

图12-6是一个简单的名词短语(NP)的扩充转移网络，该网络主要用来检查NP中数的一致性问题，其中用到的特征是Number(数)，它有两个值Singular(单数)plural(复数)。

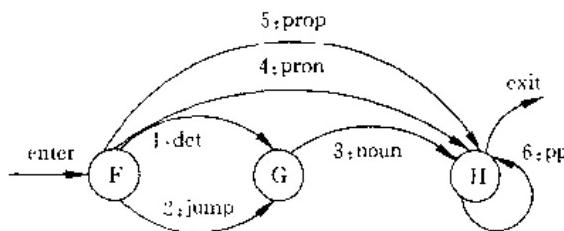


图12-6 名词短语(NP)的扩充转移网络

网络中弧上的条件和操作如下，其中C表示条件，A表示操作，\*表示系统当前正处理的词汇：

NP-1: f → [det] → g // 当前词为限定词，网络状态由f转移至g

A: Number = \* . number // 使NP的特征“数”的特征值等于当前输入限定词的特征“数”的特征值

NP-2: f → [jump] → g // 网络状态直接由f转移至g，不对应句法成分和输

### 入词汇

|                             |                                       |
|-----------------------------|---------------------------------------|
| NP-3: g→[Noun]→h            | // 当前词为名词, 网络状态由 f 转移至 g              |
| C: Number = * . Number or ∅ | // 如果当前名词的数与 NP 的数相同或者 NP 的数为空        |
| A: Number = * . number      | // 使 NP 的特征“数”的特征值等于当前输入名词的特征“数”的特征值  |
| NP-4: f→[pronoun]→h         | // 当前词为代词, 网络状态由 f 转移至 h              |
| A: Number = * . number      | // 使 NP 的特征“数”的特征值等于当前输入代词的特征“数”的特征值  |
| NP-5: f→[proper]→h          | // 当前词为专用词, 网络状态由 f 转移至 h             |
| A: Number = * . number      | // 使 NP 的特征“数”的特征值等于当前输入专用词的特征“数”的特征值 |
| NP-6: h→[pp]→h              | // 进入子网络, 介词短语网络, 本层网络状态不变, 使网络具有递归性  |

因此, this book, the books, these books 都可顺利通过上述网络, 但是 this books, these book 就无法通过。通过的名词短语的特征值都会被赋予相应的特征值。

上面给出的只是一个极其简单的网络, 用以说明 ATN 网络的基本构成和处理方法, 作为能够处理实际语言的完整的 ATN 是相当复杂的, 在实现过程中还需解决许多问题。

### 第三节 语义层：格语法

菲尔墨于 1958 年提出了格辩理论, 即通常所说的“格语法”, 主要是为了找出动词和跟它处在结构关系中的名词的语义关系, 同时也扩及动词或动词短语与其他的各种名词短语之间的关系。较之于短语结构语法, 它对于句子的深层语义有着更好的描述。无论句子的表层形式如何变化, 如主动语态变为被动语态, 陈述句变为疑问句, 肯定句变为否定句等, 其底层的语义关系, 各名词成分所代表的格关系不会发生相应地变化。

它对于句子的定义是:

$$S = M + P \quad \text{句子} \rightarrow \text{情态} + \text{命题};$$

其中  $M$  称为转换标志, 给出了从句子的整体角度而言, 描述句子的各种时态、情态、体态,  $P$  称为句子的语义模型, 表示为:

$$P \rightarrow V + C_1 + C_2 + \dots + C_n;$$

其中  $V$  代表句中的动词,  $C_i$  表示句中出现的名词或名词词组, 分别与动词构成一定的格关系, 典型的格关系如施事、受事、对象、时间、处所、工具、条件、目的、方式等。由于语义概念可以连续不断的分化与细化, 所以格名称不是一个有限集。实际系统可根据所采取的处理结构等多方面的原因进行取舍。图 12-7 中给出了用格语法描述的句子。

由此可见, 句子的结构主要由动词确定。每个动词都可定义一个格结构, 确定哪种格是必须的, 哪种格是允许的, 哪种格是禁止的, 从格语法的角度描述动词支配名词性成分的方式及语义关系, 是一个具体动词或动词类的概念的具体化。具体化的第一步是确定一个动词或动词类能和几个外部概念相联, 这就是配价概念。外部概念可以是以动词性词为中心的逻辑命题

“事件”，也可以是以名词性词为中心的逻辑概念“事物”。动词支配外部概念的个数就是价。动词支配的外部概念形式化为槽，槽以语义格的名称命名。此外，还有补充成分与动词间的联系，也称作槽，也以语义格名称命名。在使用中动词的槽有必选与可选之分，也就是说有必选格和可选格两种。必选格是句子必不可少的，如果缺失，则必须从上下文或情景语境中补上才能对句子进行理解的语义部分，如施事、受事，句法上表现为句型成分。可选格多为工具、方式、目的、时间、处所、原因、条件等，由附加成分构成。如“创作”的语义框架为

〈创作,施事格\*,受事格\*,时间,处所〉,

其中带星号者为必选格，其余为可选格。“施事格”中应填充“作者”，“受事格”中应填充“书籍”。

## 第四节 语用层：篇章语法

前几节中讨论的各种理论与实现方法主要是着眼于单句，这一节中将转向由一个或多个句子、段落组成的文本，探讨它不同于单句的特点和在这一领域中的典型理论。

### 一、篇章结构与文本连贯性

文本由单句组成，但不等同于句子的简单叠加。它具有特定的篇章结构，即文本的各部分之间存在的关系必须能够保证文本语义的连贯。

从语言学角度而言，篇章的结构特征是区分篇章是否完整，区分不同的体裁形式或语域（在特定领域中使用的语言，如数学领域、诗歌领域）的基本原则。通常而言，人们在写作或组织话语时，对于应采用何种手段，这些手段怎样衔接形成篇章，都会遵守一些约定的、普遍接受的方式，也就是“篇章结构”。

例如：科技语体结构：

|             |         |
|-------------|---------|
| REPORT      | 报告的目的   |
| SUGGEST     | 对研究提出建议 |
| PRESENT     | 介绍研究步骤  |
| DESCRIBE    | 叙述结果    |
| DEMONSTRATE | 显示图表    |
| SHOW        | 说明意义    |

科技论文的结构大致如上所示，即“概括性的结构定式”，作者根据不同的具体需要，确定所需的若干部分和每部分的实际内容，即选择“实际的意义结构”。

文本连贯指使用语言形式进行交际的过程中，目标和概念上的统一。从读者的角度而言，就是能够根据上下文来掌握作者的意图，不产生混淆或错误。研究篇章结构的目的就是为了保证生成文本在语义上的连贯性，即采取何种方式保证所生成的文本中的每部分都通过一定的方式与整体相关。

目前，描述篇章结构的代表性方法，也即篇章语法，主要有两种：框架理论和修辞结构理论。框架理论是 Kathy McKeown 在宾夕法尼亚大学开发 TEXT 系统时提出的，之后在哥伦比亚大学继续进行这方面的研究。修辞结构理论首先由 Sandra Thompson, Christian Matthiessen 和 William C. Mann 提出的，主要研究基地在南加州大学信息科学研究所。

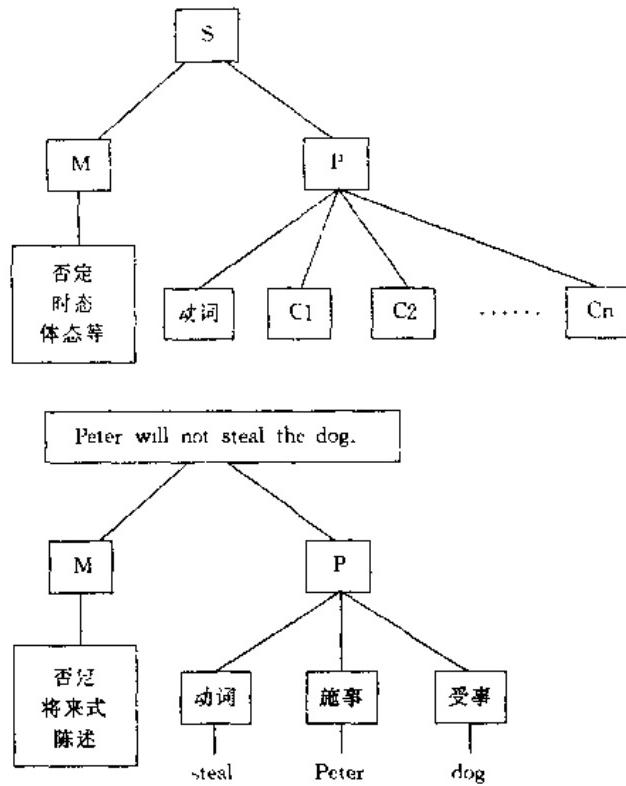


图 12-7 用格语法表示语句

## 二、框架理论

第一个明确地引入篇章结构的生成系统是 TEXT84。它把篇章结构的基本组成因素定义为“修辞谓词”，修辞谓词间的排列次序和联接关系是固定于系统中的，通过这种人为的规定保证文本的连贯性。这种方法没有明确提出关于文本成分间联接关系的理论。

该系统使用了“框架”，预定义了典型的段落结构，以控制段落中句子的内容与顺序。框架由一系列的修辞谓词组成，它通过限定输入概念必须包含的语义内容来选择可填入的概念，其灵活性体现在框架本身是可嵌套的，以及依据输入概念的要求，部分修辞谓词可被忽略或重复出现。本质上，这种框架等价于段落模板。规划模块对模板进行选择或通过嵌套动态构造新的模板，并依据知识库内容对模板空槽进行填充。

### 1. 修辞谓词

修辞谓词，是作者或说话者用以保证篇章连贯性的手段，是可选用的传递信息的语言方式。例如，“类比”：通过对不同事物共同点进行描述，传递当前对象的特征信息。对于这些修辞方法的语言学研究表明，构成篇章时，某些修辞谓词的组合和相对次序更易为人们接受，而另一些组合和相对次序就会引起意义的含糊甚至误解。

例如，描述对象时，经常采用的修辞方法是：1) 声明对象属于某个类别；2) 描述对象的功能、属性、组成等；3) 类比描述某个与其有共同点的，更为人们所熟识的对象；4) 举例说明；而且使用次序也遵守一定的规律，通常声明对象所属的类别在前，举例说明在后。

### 2. TEXT 系统

TEXT 系统定义了两种与篇章结构有关的对象：谓词语义和框架。

谓词语义实质上是一种特殊的模板化匹配函数，定义了谓词以何种方式在数据库或知识库中获得数据，其参数的类型和数目取决于该谓词的实际语言学语义。谓词语义仅适用于 TEXT 系统，如果用于另一个系统，就必须重新定义。匹配时，它的参数可取具有相同类型的任何值。下面给出一个“标识”谓词的语义模板：

给定参数：实体

(标识(实体)(父结点)  
(限定(属性名)(属性值)  
非限定(属性名)(属性值))  
)

举例：

(标识 SHIP WATER—VEHICLE  
(限定 TRAVEL—MODE SURFACE  
非限定 TRAVEL—MEDIUM WATER)  
)

模板中，“实体”表示要求给定的参数，“父结点”表示给定“实体”在知识库层次中的上层结点，“属性”表示实体的属性，“限定”行表示要求找到专门属于此类“实体”的那些属性，与之相反，“非限定”行表示要求找到并非专门属于此类“实体”的那些属性。举例表示给定了“SHIP”为“实体”参数后，模板匹配结果。

TEXT 系统中框架完成的功能是确定谓词表达式的结合方式，以形成完整的文本。

例如，描述客观对象的“标识框架”：

标识符(类别 & 属性或功能)  
{类比/组成/属性}\*  
举例/证明+  
{引伸/类比/属性}  
{举例/证明}

其中，圆括号中的评论指出“标识符”框架可以说明实体所属的类别，具有的属性或可完成的功能；花括号中给出的“类比、组成、属性、引伸、举例、证明”均为可选的谓词，分别对应于各自的语义匹配模板，只能依据系统当前的知识库状态从中选取一个谓词；\*号表示可以有零次或一次重复；+号表示至少有一次重复。

框架高度抽象出作者可使用的手段，通用的排列次序，构造完成一定通信任务的标准篇章结构模式。它用于导向生成过程，选择“说什么”和“什么时候说”。它从完成同样功能的文本中人工分析抽取，一些经常使用的模式被抽象出来并以框架表示。由于实际文本的结构模式是非常松散的，因此，在模式的每个判定点上，都有许多修辞谓词可供选择，同一种模式可衍生出多种不同的结构，具有一定的灵活性。

从篇章结构的角度来看，TEXT 系统具有以下特点：

- (1) 框架的范围是整个文本；文本的结构是事先定义的，而非动态生成的；
- (2) 文本中的可选项是就整个文本而言；
- (3) 文本中各组成部分的顺序是预定义的。

### 3. 系统实现

篇章结构决定于说话者的目标和他想说的内容,因此,框架的选择也就取决于系统当前应完成的通信目标和与该目标相关的知识结构。每个系统内预定义的通信目标都有若干框架与之相联,之后依据系统知识库中与该通信目标有关的内容(相关知识基),即系统中存在的与该目标具有潜在相关性的知识的数量和类型,从中选择一个框架。

框架的形式化表述类似于一个扩展转移网络(ATN)。经过它的弧时,发生如下动作,从相关知识基得到所需信息,从中选择输出文本中的一个命题<sup>①</sup>,并确定框架的当前状态。在扩展转移网络的每一分支点上,所有可能的转移状态都会被计算,并由系统设定的选择函数从中选择一个最佳方案。框架填充后得到的命题集合,送入系统实现模块,得到最终的输出文本。

#### 4. 存在问题

尽管框架结构仍旧是文本生成系统中广泛使用的方法,但它有着难以克服的弱点:缺少对每部分在总体结构中通信目标的表示;当文本生成的某个部分未达到预期效果时无法重新规划,系统不能动态地规划生成文本中各语义单元次序;无法精确定义谓词的语义,对谓词语义的解释取决于不同的系统。

### 三、修辞结构理论

#### 1. 理论提出

新的篇章结构理论是修辞结构理论,它提出的修辞关系,用于涵盖篇章各部分间的关系。典型的实验系统是 NOAH90,采用了这种理论和自顶向下逐级细化的结构。

1987,Mann 和 Thomson 经过对于上百篇不同类型的文本研究之后,提出了 20 种存在于一般性英语文本(不包括小说和诗歌)相邻部分间的关系。这些修辞关系,在文本结构中可以被递归使用,称之为“修辞结构理论”(RST)。这种理论的假设前提是,文本被认为是连贯的,当且仅当其所有组成部分都能够通过递归使用修辞关系最终归纳入一个涵盖全部文本的总体关系中。它完全是描述性的,没有对于其修辞关系进行形式化描述,也没有给出其关系集合的完整性证明。不过,它的确包含了 Hobbs 提出的绝大多数关系并且支持 Makeown 的框架理论。

#### 2. 基本元素

修辞结构理论中定义的对象:关系、模式、模式应用、RST 结构。

##### (1) 关系

一个关系的定义分为四个部分:对核心部分的限制条件;对附属部分的限制条件;对核心部分和附属部分相联的限制条件、效果。只有这些限制条件满足,两段文本间的修辞关系才能成立,这些条件可以为空。“效果”表明说话者或作者在使用此关系连接两段文本时希望达到的目标,它不会为空。

例如对于“证明”这个关系的定义如下所示:

对核心部分(论点)的限制条件:

读者可能并不相信此论点;

对附属部分(论据)的限制条件:

读者相信论据或者可能相信它;

对核心部分和附属部分相联的限制条件:

<sup>①</sup> 命题是修辞谓词的一次实现。谓词是系统预先存储的模板,命题则是由生成过程中动态生成的。

读者对于论据的理解可以增加他对论点的相信程度；  
效果：  
读者对于论点的相信程度增加了。

### (2) 模式

图 12-8 以图形的方法直观描述了一个基本 RST 模式。一段文本，以整个图形表示，被分为两部分，分别称为“核心”部分和“附属”部分，两者间通过特定关系相联。图 12-9 给出了一些常见的模式，有的仅包含一个关系，也有的包含两个，当然也可以包含多个关系。图中的“详述”和“解决方式”两种模式，“核心”部分和“附属”部分的相对顺序正好相反，这似乎表明模式预先规定了“核心”部分和“附属”部分的相对次序，事实上并非如此。在模式定义中，次序并未确定，图示中只是给出最常见的次序。例如，“详述”的附属部分通常跟随其核心部分，而“解决方式”的附属部分（描述问题）通常出现在其核心部分之前。

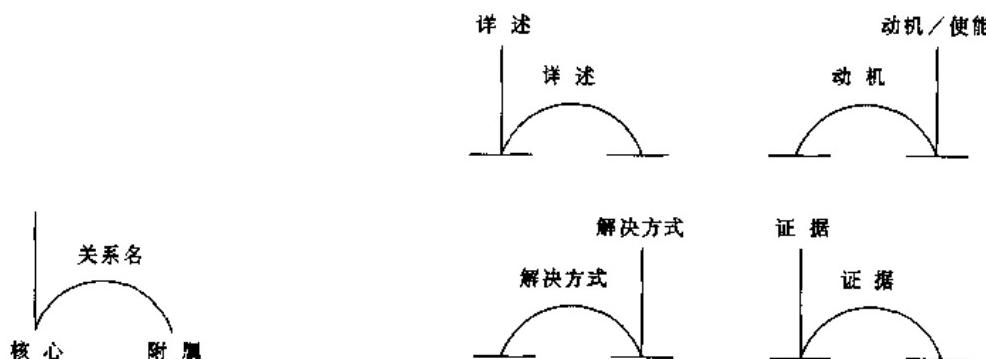


图 12-8 基本的 RST 模式

图 12-9 常见关系模式

### (3) 模式应用

指模式在当前生成任务中的实例化。

### (4) RST 结构

RST 理论下的篇章结构是多个模式应用的组合，如图 12-10 所示。

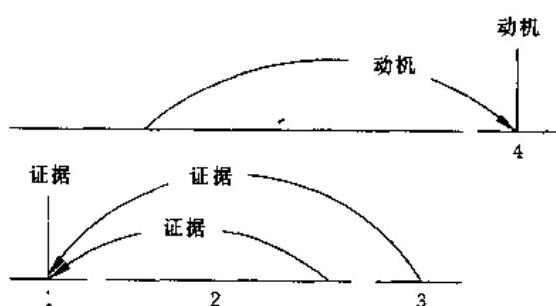


图 12-10 RST 理论篇章结构

整个文本由 4 个子句级概念组成。在图的最顶部，是“动机”模式的一个应用，其核心部分包括子句 4，其余三个子句组成附属部分，两者间具有“使能”关系，附属部分应完成的目标是使读者愿意或者能够完成核心部分所要求的动作。由于核心部分仅包括一个子句，不能再被继续分解。附属部分继续分解为“证据”模式的一个应用，其核心部分为子句 1，有两个附属部分：子句 2 和 3，分别与核心部分形成“证据关系”。由上例可知，模式可用于分解任意大小的文本部分，可递归使用。这种自顶向下的分解方式最终得到一个树形结构，递归描述整个文本各部

分间存在的修辞关系。

### 3. 修辞结构理论的形式化描述

利用修辞结构理论描述的修辞篇章结构,称为修辞篇章结构,可为规划模块<sup>1)</sup>提供一种限制搜索空间的方式,只有候选概念满足核心部分和附属部分的限制条件,才能作为对应于该部分输出位置的候选人。这种修辞篇章结构可形式化为一个树形结构,所有的叶子表示待输出的内容。每个子结点形式上可表述为一个五元组{名称,效果,限制,前提,分划},其中:

- 名称:修辞篇章结构的特定标识;
- 效果:修辞篇章结构应达到的一个或多个通信目标,即针对阅读者状态确定的,篇章的作者所希望达到的效果。这些目标是表达说话者的特定意图,可表示为用户状态模型的变化;
- 限制:使用修辞篇章结构之前,知识库中相关知识基及用户模型必须满足的条件;
- 前提:为保证修辞篇章结构的连贯性与一致性,相关知识基及用户模型应满足的条件;
- 分划:一组有序的子语义群序列,其顺序反映了篇章连贯性的要求。每个子结构若标注为“可选”,则在恰当情况下可忽略此结构;若标注为“必须”,则代表着篇章的主干。一般而言,子语义群分为下列两类:上级语义群目标分解得到的通信目标,如“说服”可分解为“激励”或“表述”,需要再次进行分划;限定于词组或小句范围内的语义,可直接生成为句子,如传达、询问、命令等。

### 4. 形式化实现

规划器的工作流程是自顶向下进行依次扩展,具体而言:

- (1)给定一个或多个通信目标,它找到合适的 RST 关系,其效果部分能够满足全部或部分目标;
- (2)对于每个修辞关系,确定输入语义结构中哪些能够满足核心部分的限制条件;
- (3)未被满足的限制条件作为子目标,成为系统下一级规划的起点;规划树以先深搜索或先宽搜索的方式扩展;
- (4)当遇到下列情况时,规划器终止:
  - 相关知识基中的所有语义结构已经用完而仍有未完成的子目标,此时系统可要求更多的符合特定条件的输入;
  - 所有通信目标均得到了满足。
- (5)实现模块按从左到右的顺序遍历生成的规划树,对规划树底层结点生成相应的子句,其分支点生成典型的表示子句间关系的词或短语。

### 5. 存在问题

这种生成方式的缺点是:

- (1)没有恰当的机制,确定句子的范围;
- (2)把语义输入结构视为无法分割的原予体,无法对输入的语义结构进行合并或扩展。
- (3)由于在修辞篇章结构的起点只能是一种修辞模式,即整个文本只能由两部分组成,因此这种理论还无法适用于较长的文本。
- (4)从语言学角度出发,许多学者提出了自己的修辞关系体系,其中的修辞关系从两个到一百多个,不一而足。对实际篇章研究越细,就会发现越多的语言现象无法被目前提出的修辞关系所涵盖。

<sup>1)</sup> 在“生成系统结构”中定义。

## 第五节 自然语言生成

### 一、自然语言生成

自然语言生成就是把计算机内部以某种形式存放的需要交流的信息,以自然语言的形式表达出来。一般包括以下两个部分:

- 1)建立一种结构,以表达出需要交流的信息。也就是进行“构思”,确定要“说”的内容。
- 2)经适当的词汇和一定的句法规则,把要交流的信息以句子的形式表达出来。

目前开发的语言生成系统多是面向特定系统的输出接口,作为附属的模块集成入大型的系统中,如专家系统、机器翻译、数据库操作等,或是研究机构中用于验证各种生成理论的实验系统,投入实用的尚不多见。目前的研究重点已由独立段落的生成逐渐发展为多段落文本的生成,进而趋向于多语种生成,图表、文字等多种形式并存的复杂文档的生成等。它已经成为一个独立的研究领域,吸引了众多的研究人员。

它可用于大众服务,有统计报告生成,人员招聘启示生成;用于商业服务,有用户服务信件生成;用于专家系统,可作为咨询系统或决策系统的解释器,明确表述专家得出结论的准确步骤;用于数据库接口和其他应用,如技术报告生成,教学系统等。随着语言生成理论和技术的不断完善,语言生成系统可以集成到任何主要的计算机应用程序中,作为系统输出方式的一种,具有广阔的实用前景。

### 二、语言生成系统特点分析

#### 1. 语言理解与语言生成

与语言理解相比,语言生成就年轻得多,有着自身独到的特点和要求。

自然语言理解通过对于给定文本中提供的各种语言现象进行解释,确定语句的意义与写作者的意图,理解型系统的处理对象是人类实际生活中使用的自然语言,必须面对无穷尽的语言现象。通过在给定文本蕴含的所有可能歧义中选择完成对文本的正确理解,是被动过程。

与之相反,语言生成的核心任务是“说什么”和“怎么说”,是一个主动过程。从系统的当前目标和知识库到可能的篇章内容与组织结构是一对多的关系,从已知的语义表达达到符合语法规范的字符串也是一对多的关系,如何在可能的选项中识别出符合当前语言环境和语法规范的选项,怎样判定某个选项在当前的语言环境中优于其他选项,是整个系统设计的核心。

#### 2. 单句生成与文本生成

文本生成与简单的多个句子,如对话中的多个单句的生成有很大不同。

文本是由一个或多个句子、段落组成,但绝不等同于句子的单独生成、简单叠加,句与句之间存在概念上的连贯和语法结构上的衔接。文本的连贯与衔接是文本生成系统设计中的核心问题,是输出文本能否被接受的关键。这是篇章生成不同于单句生成的基本点。

#### 3. 书面文本与口语化文本

目前的生成系统几乎都把任务范围限制为书面文本的生成。这意味着许多在口语中常见的现象,如经常性的插入、中断、重复、大量不完整不符合语法规范的句子的使用,以及各种方

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

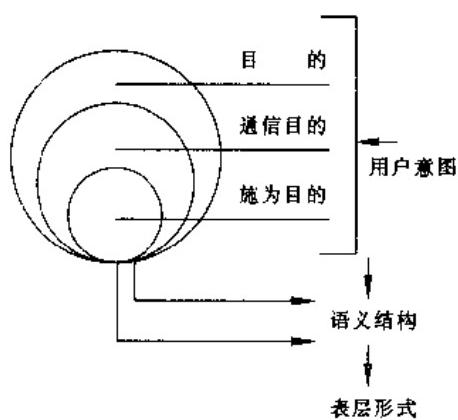


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

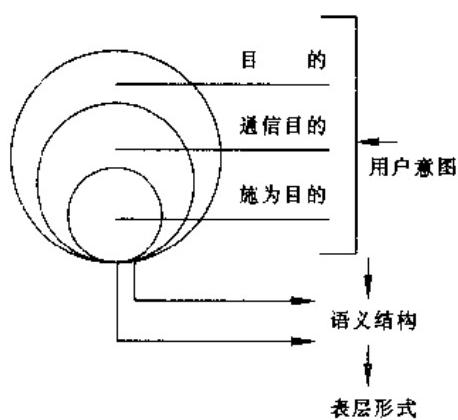


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

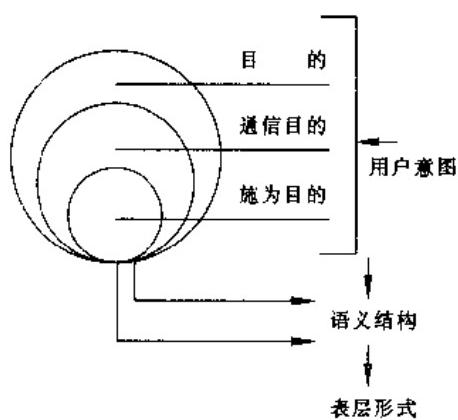


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

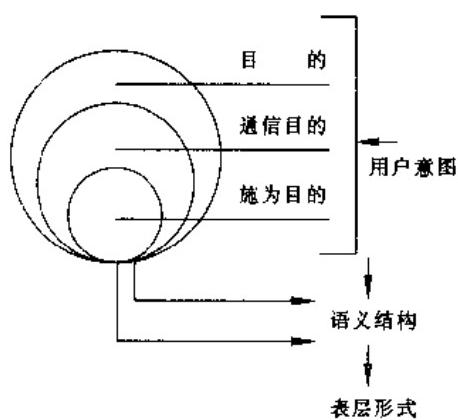


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削减冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

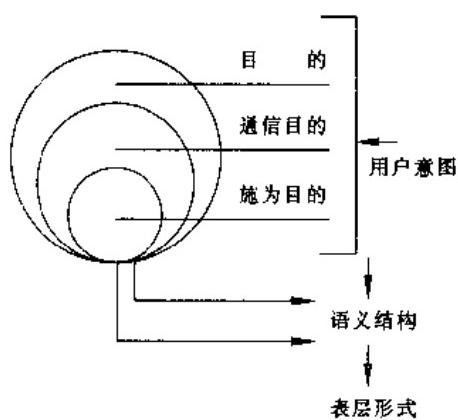


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

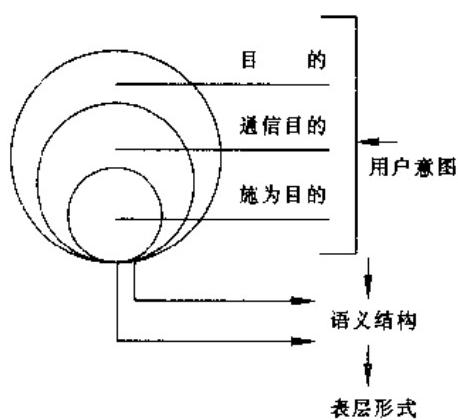


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削减冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

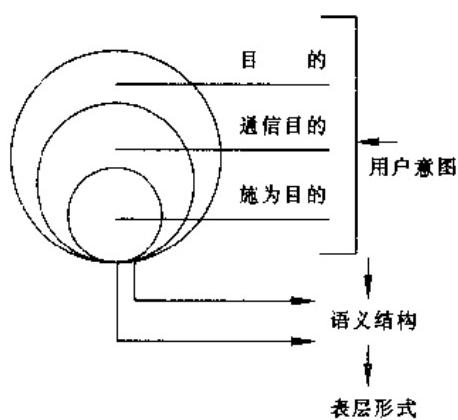


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

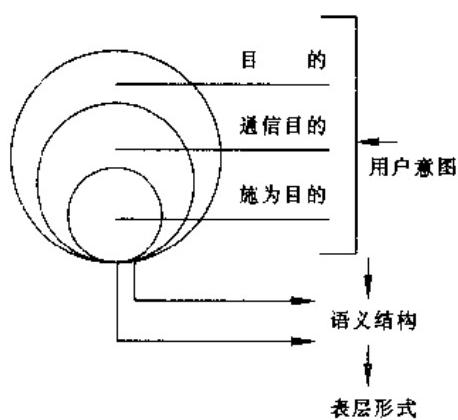


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削减冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

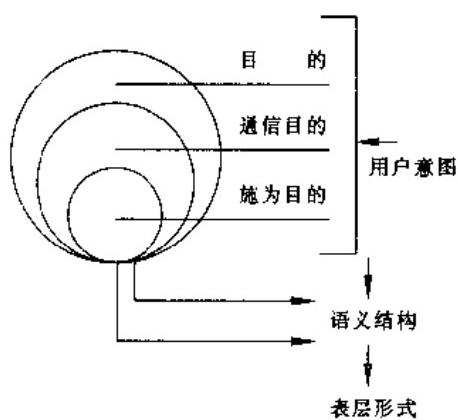


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

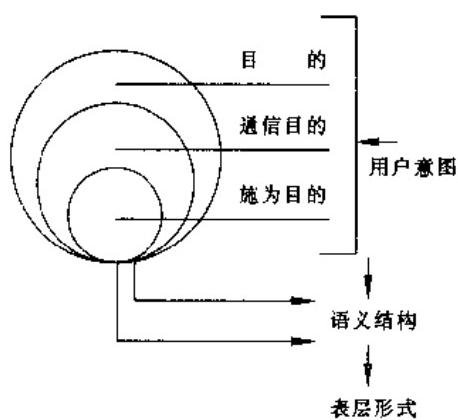


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削减冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

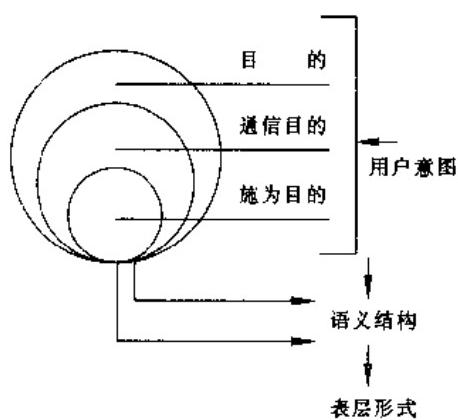


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

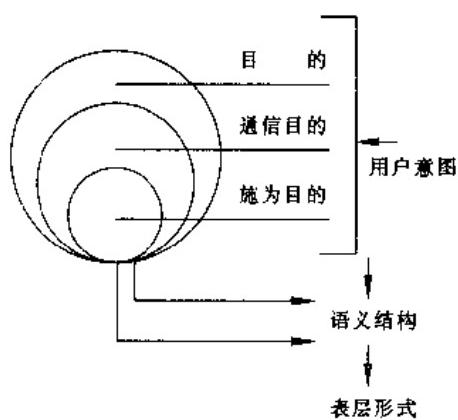


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

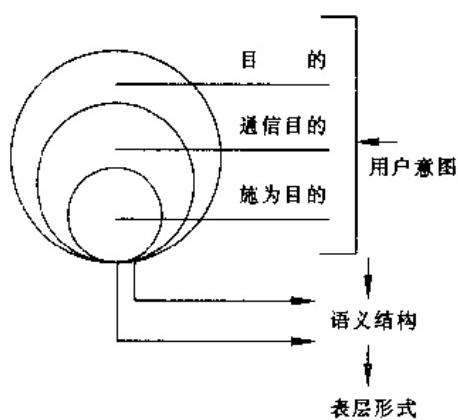


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

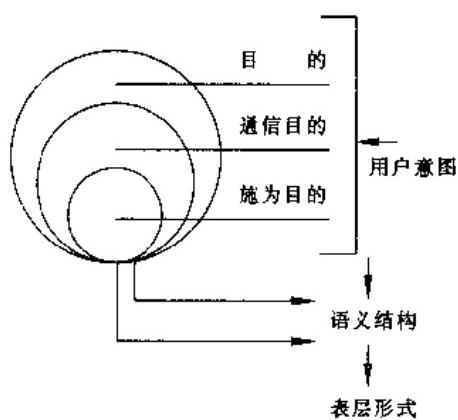


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

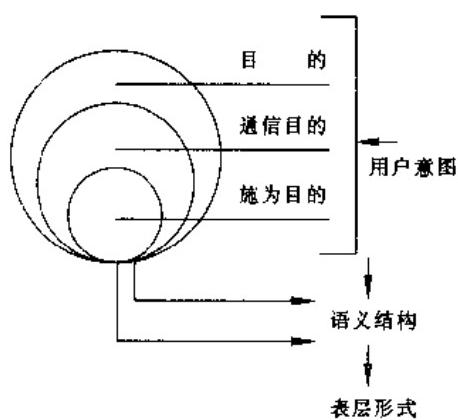


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

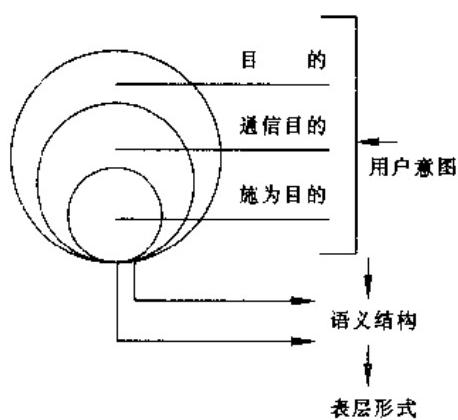


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

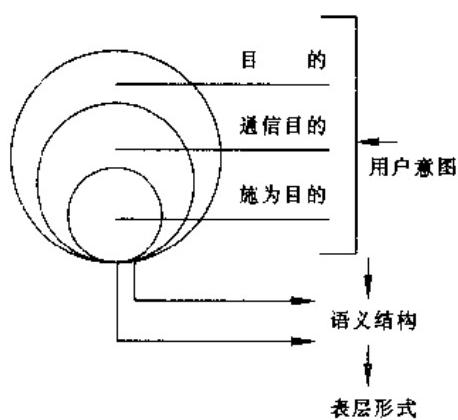


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

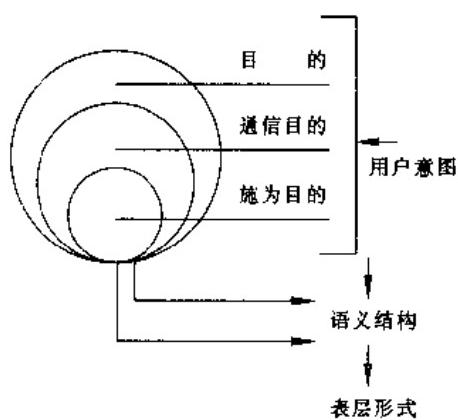


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

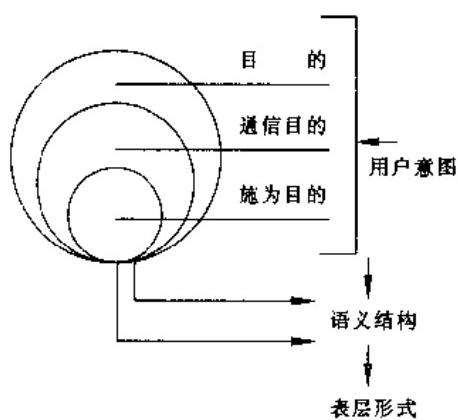


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

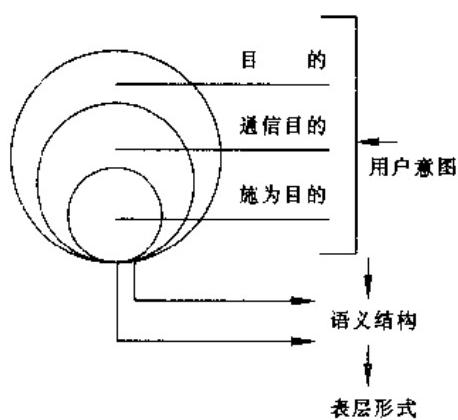


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

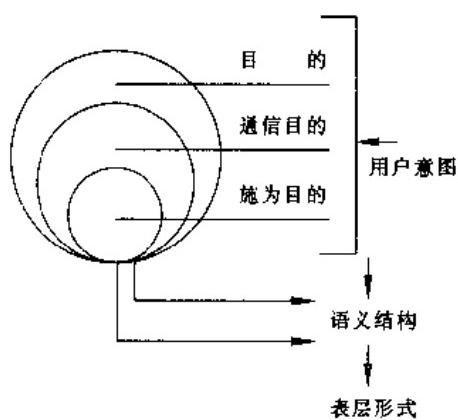


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

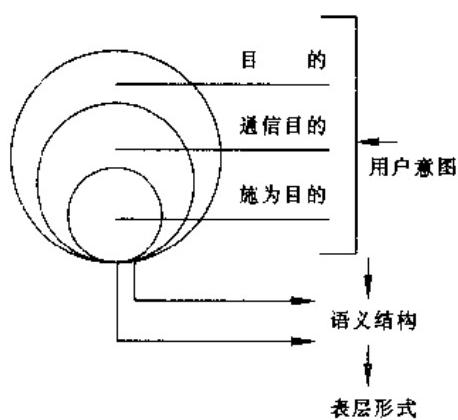


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削减冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

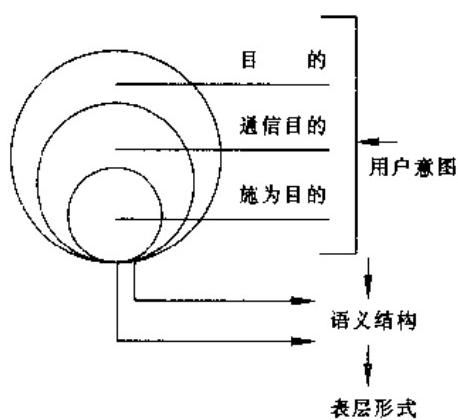


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

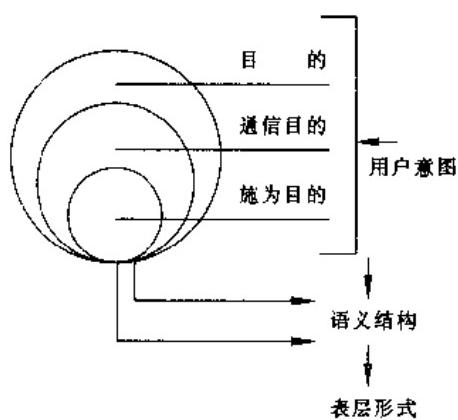


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削减冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

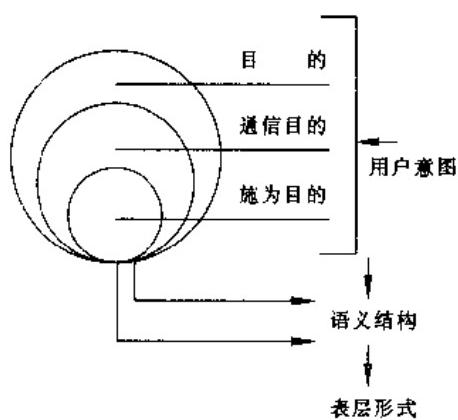


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

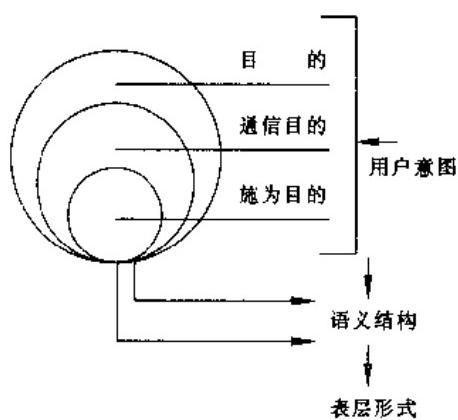


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

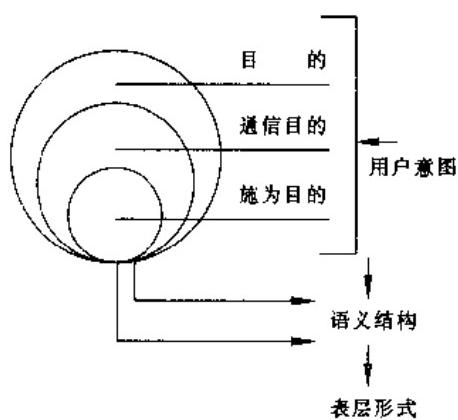


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

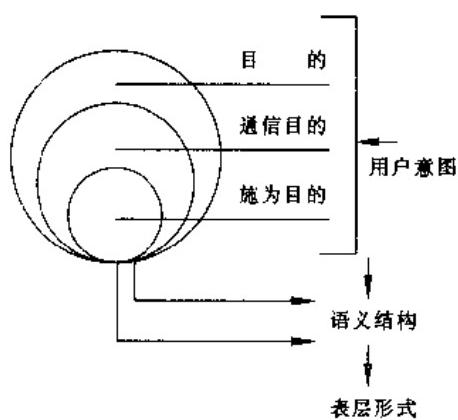


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

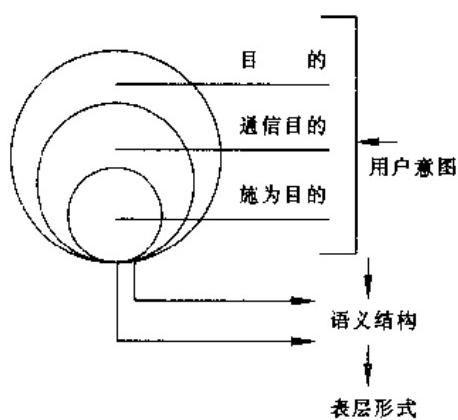


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削减冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

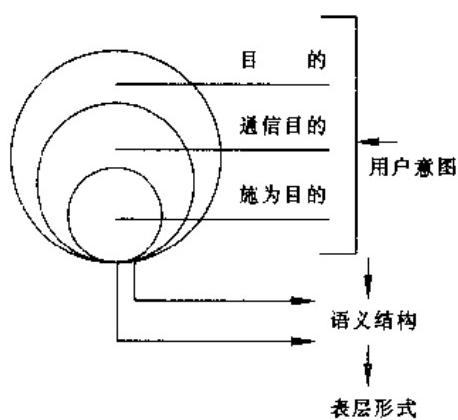


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

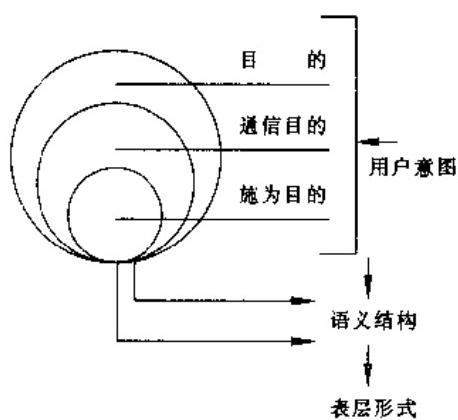


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

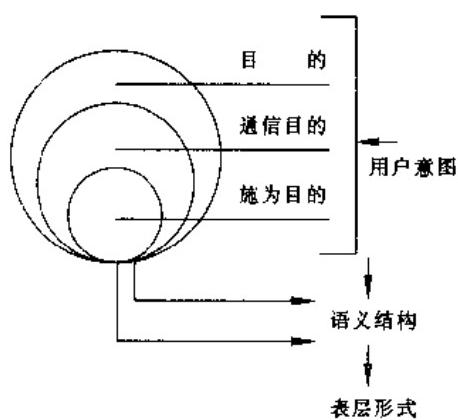


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

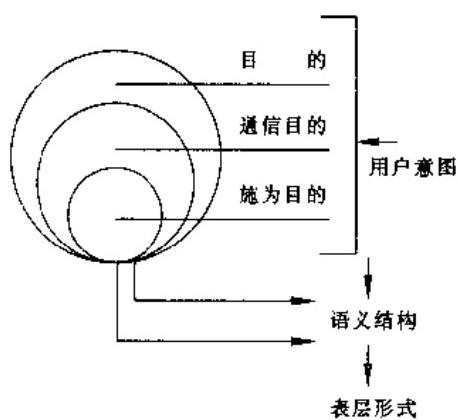


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

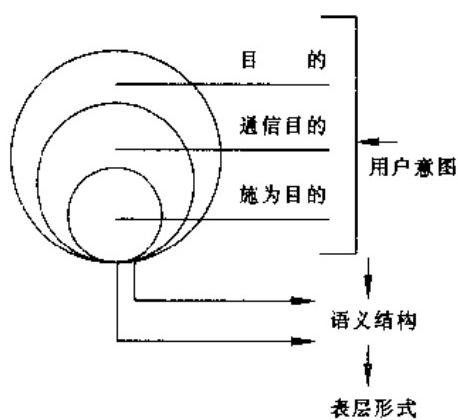


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

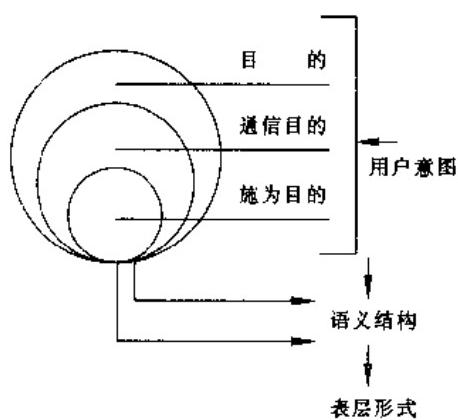


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

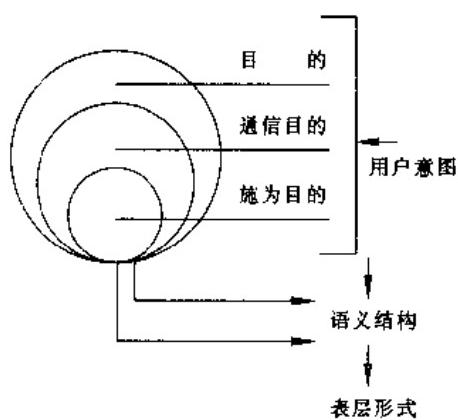


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

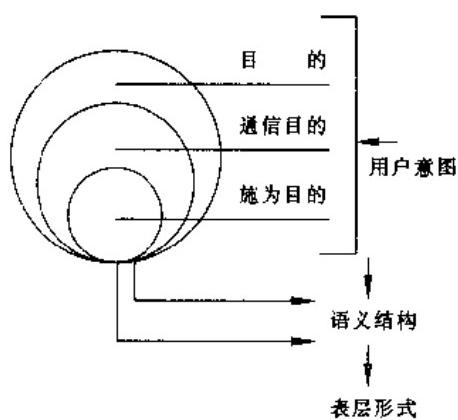


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

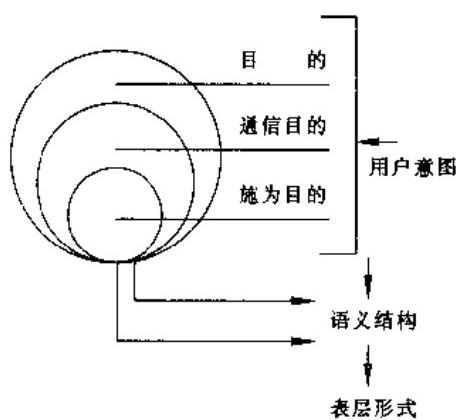


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

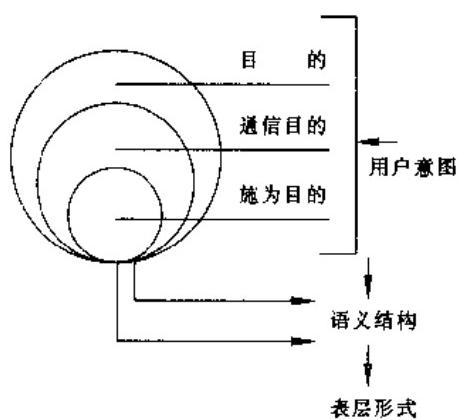


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

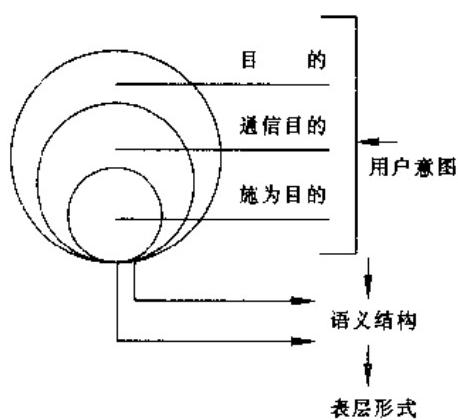


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削减冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

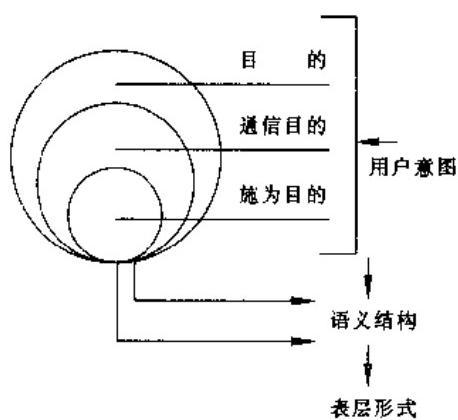


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

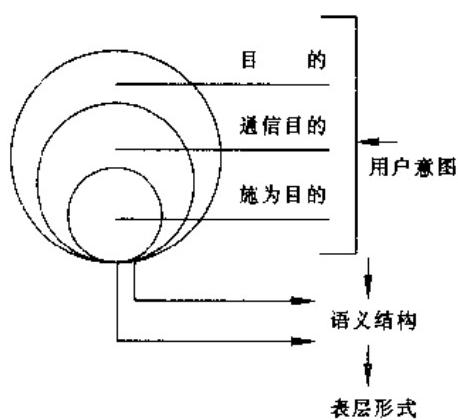


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削减冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

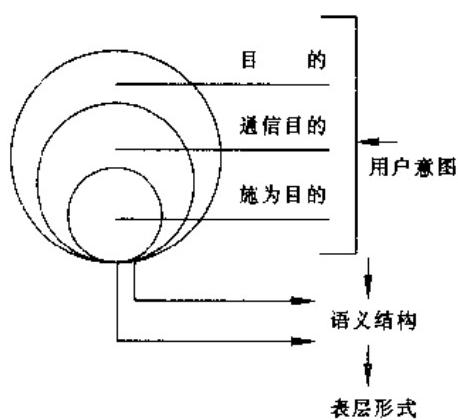


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

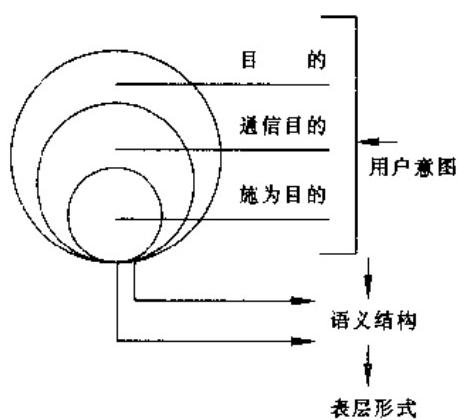


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

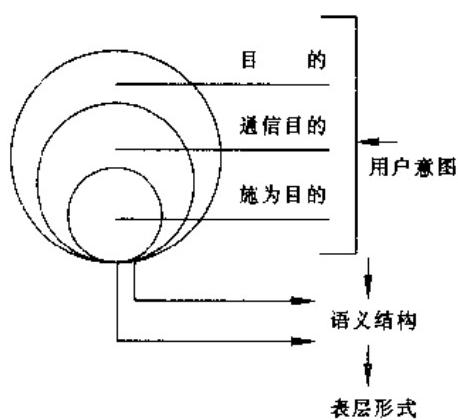


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

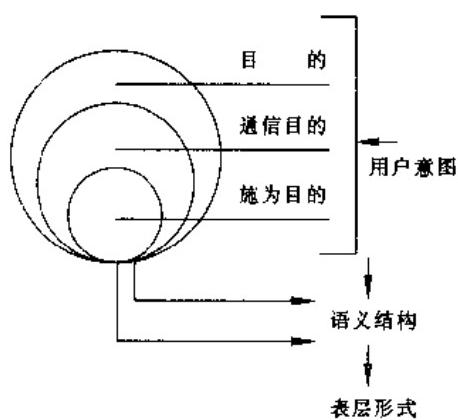


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

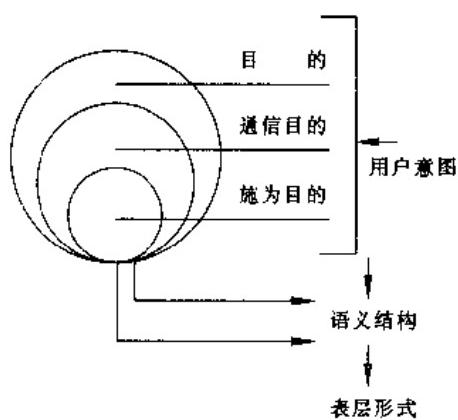


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

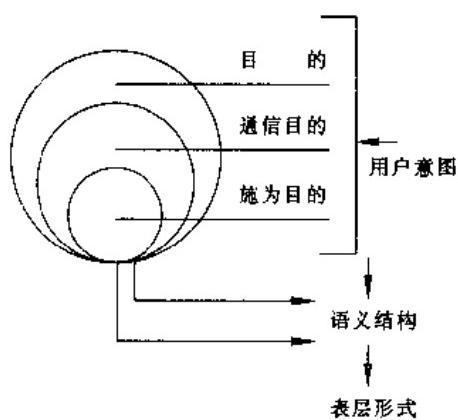


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

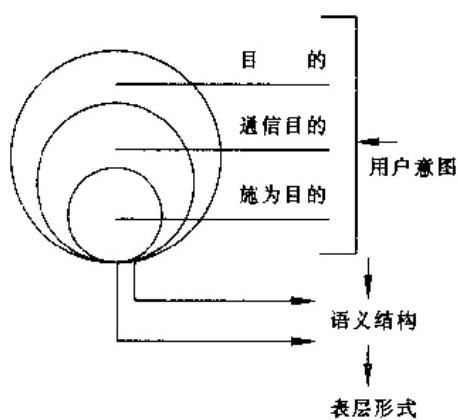


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

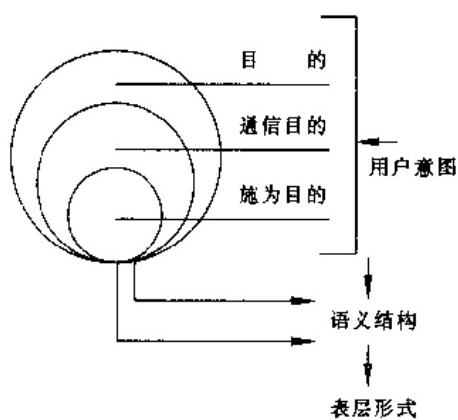


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

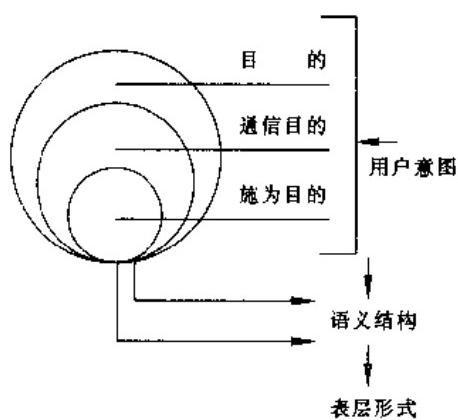


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

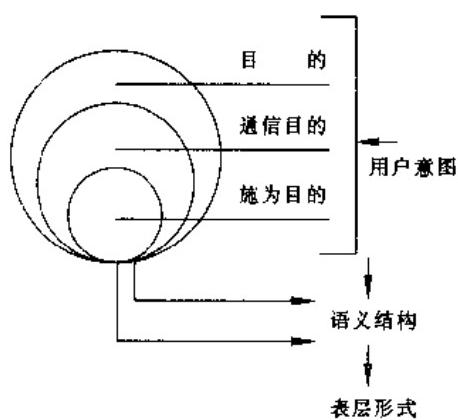


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削减冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

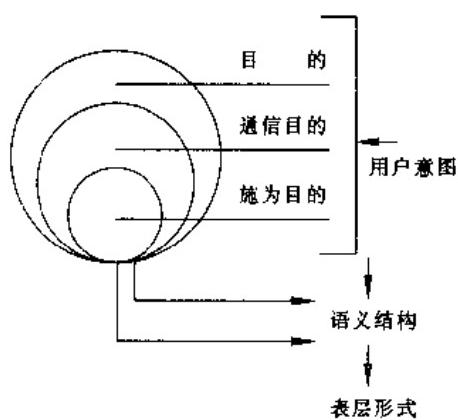


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

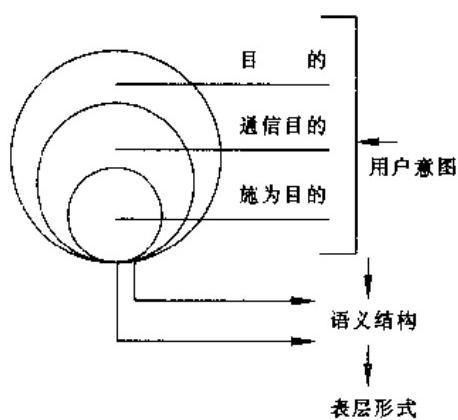


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

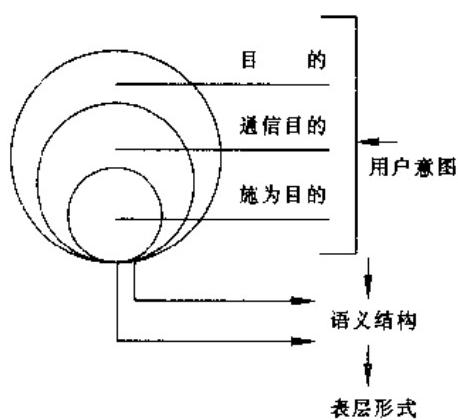


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

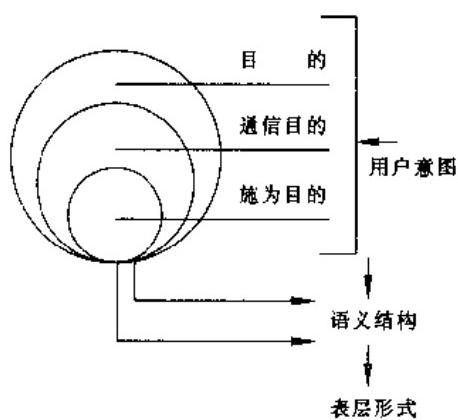


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

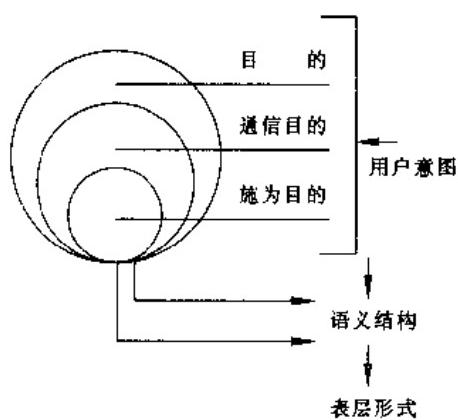


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

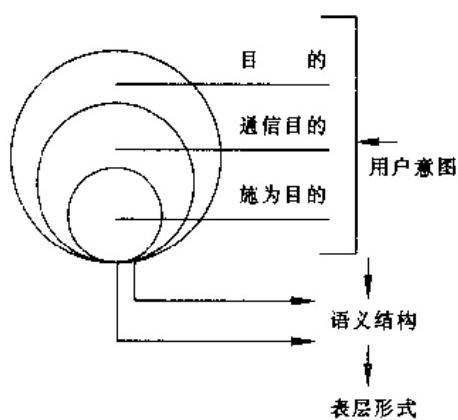


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

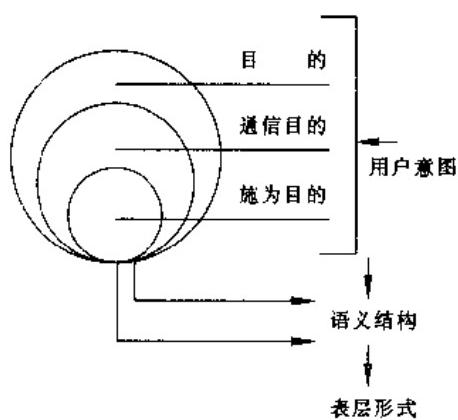


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

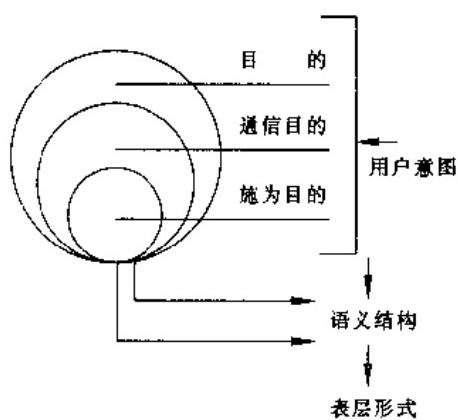


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

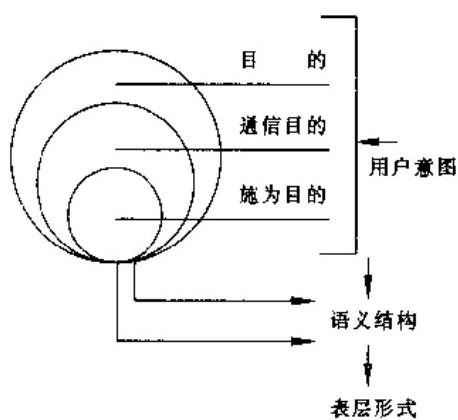


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

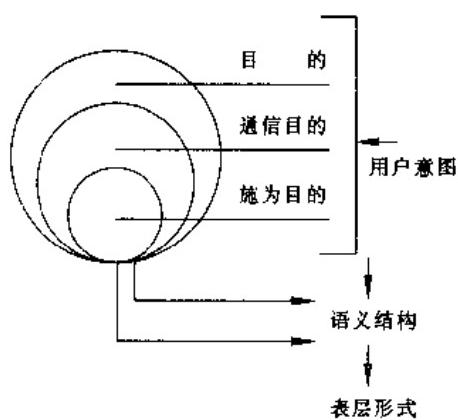


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

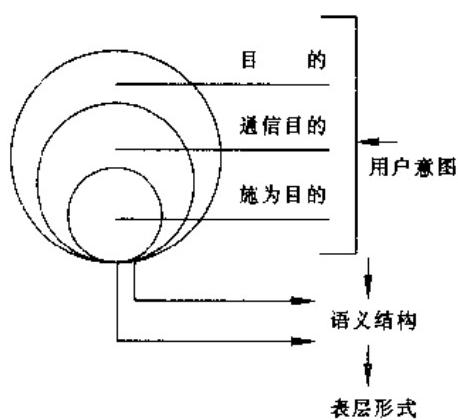


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削减冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

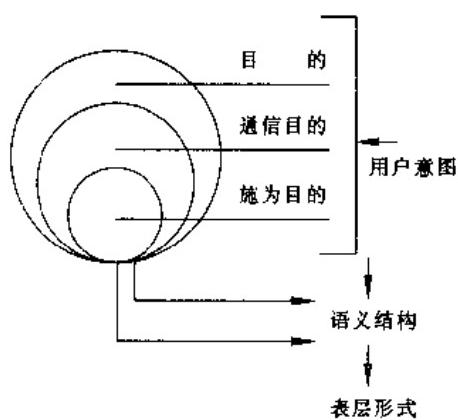


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

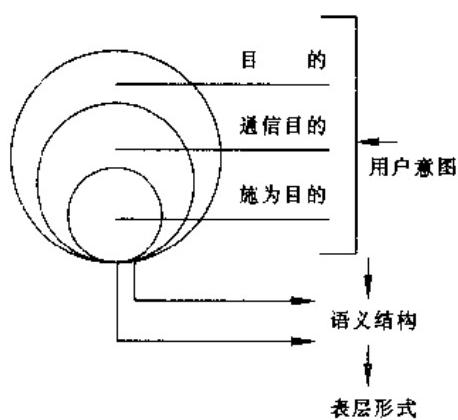


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

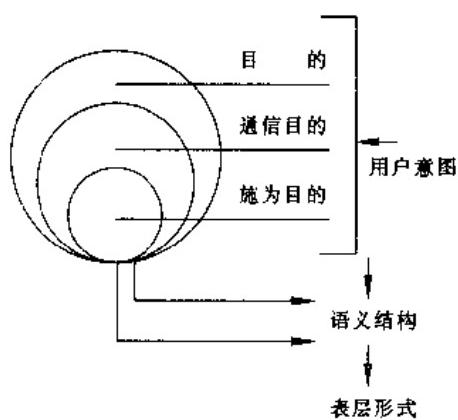


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

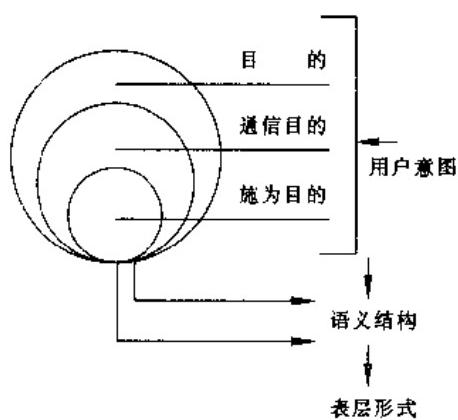


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

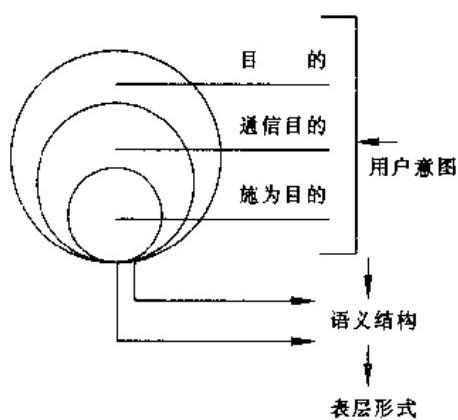


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

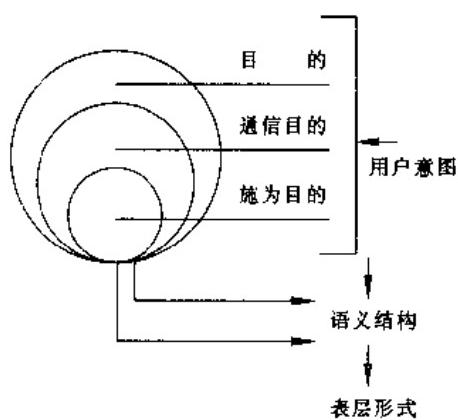


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削减冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

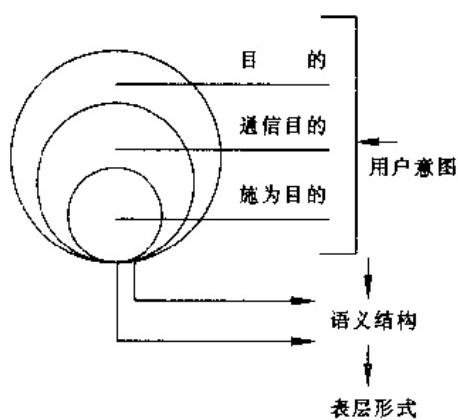


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

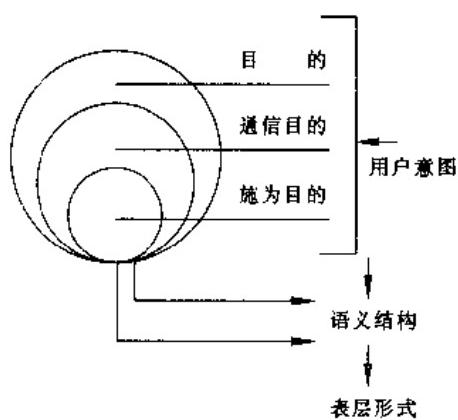


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

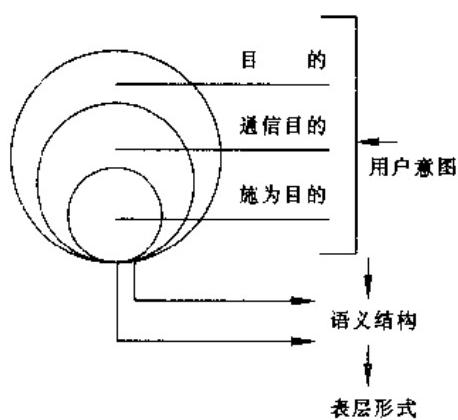


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

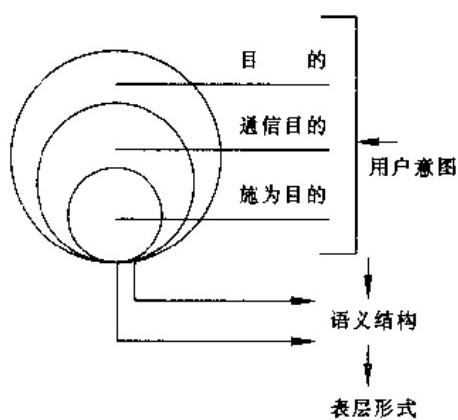


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

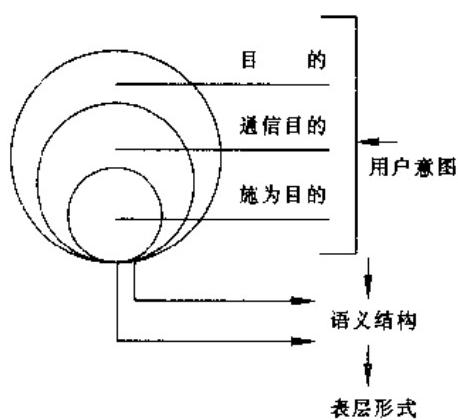


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

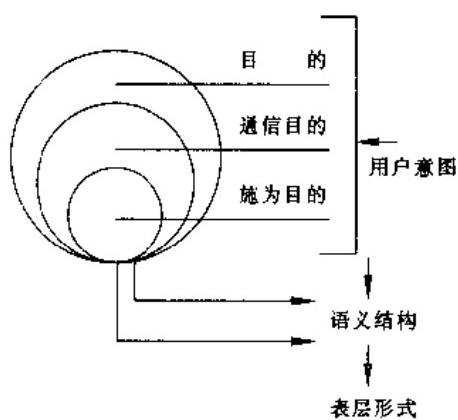


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

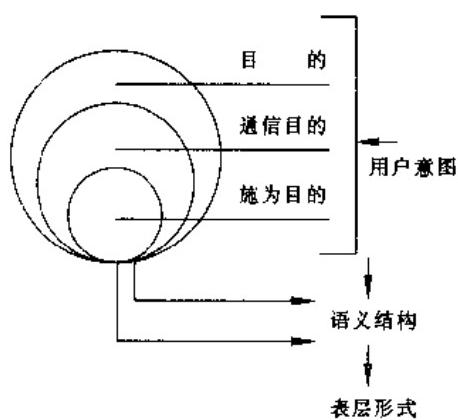


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

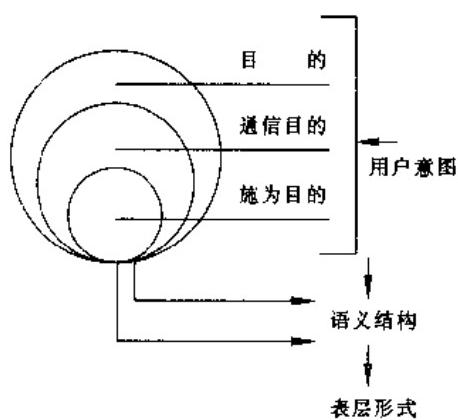


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

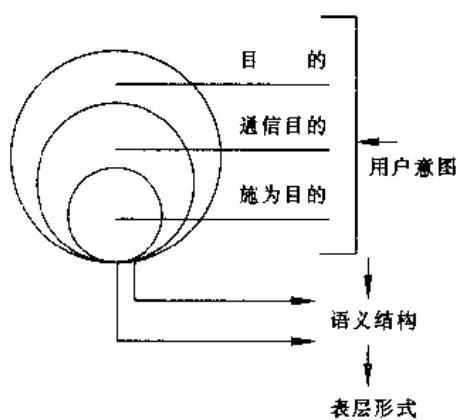


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

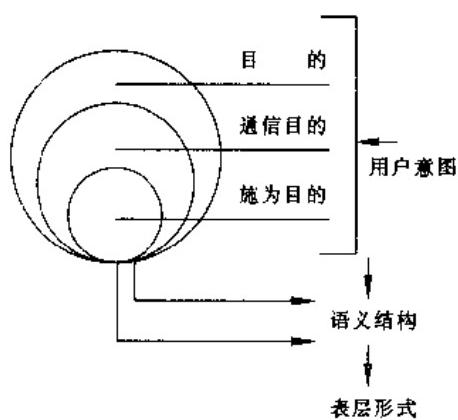


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

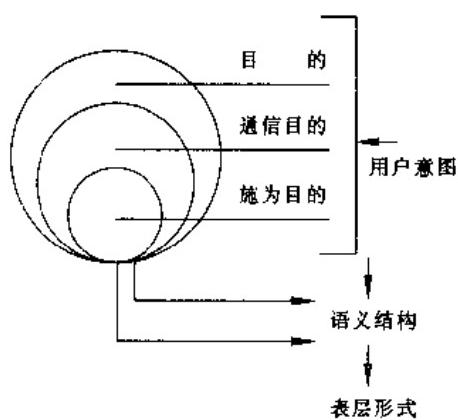


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

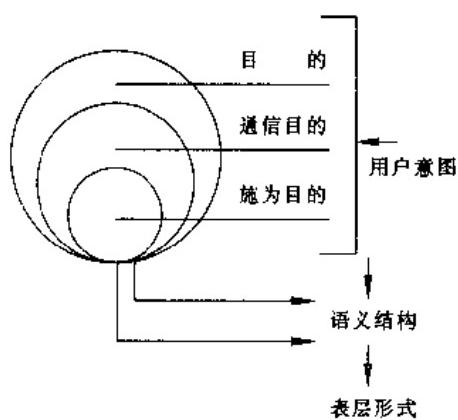


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

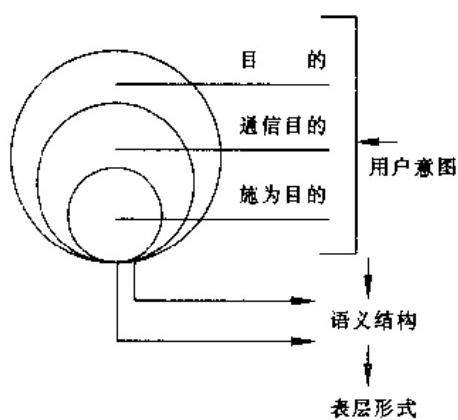


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削减冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

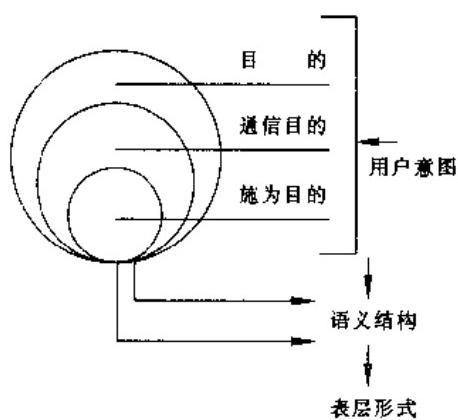


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

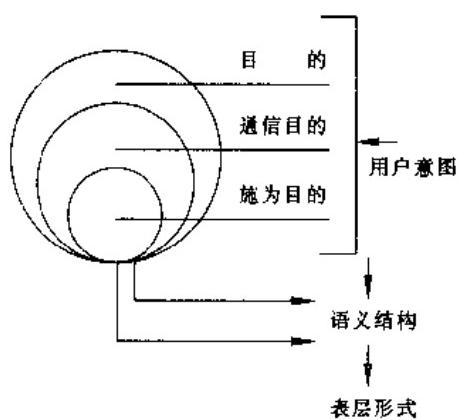


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

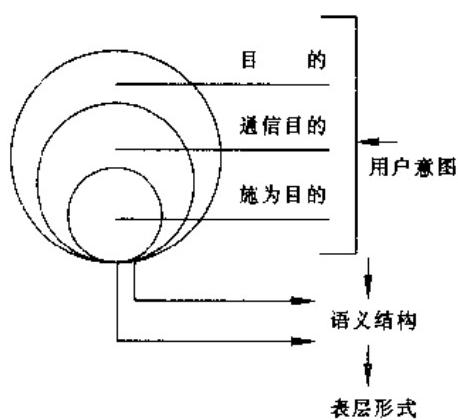


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

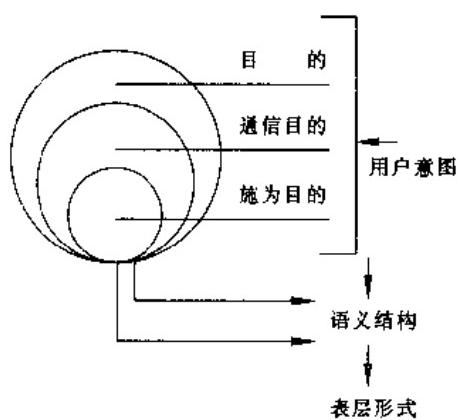


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

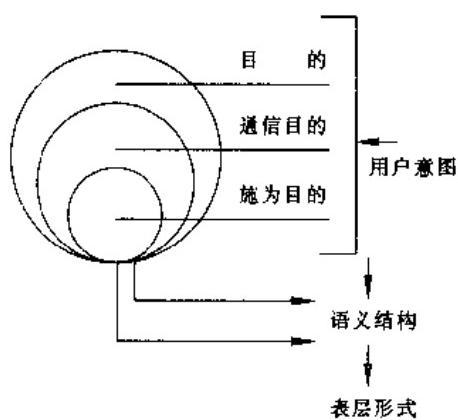


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

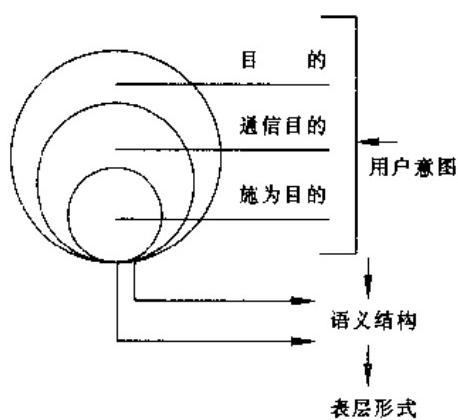


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

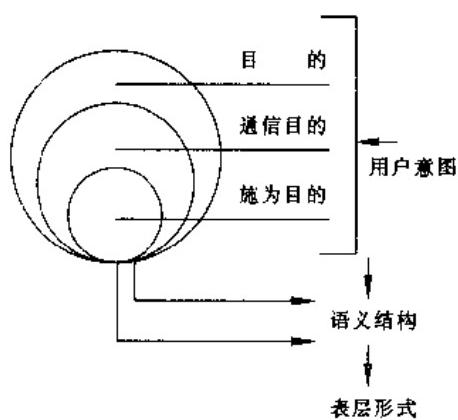


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

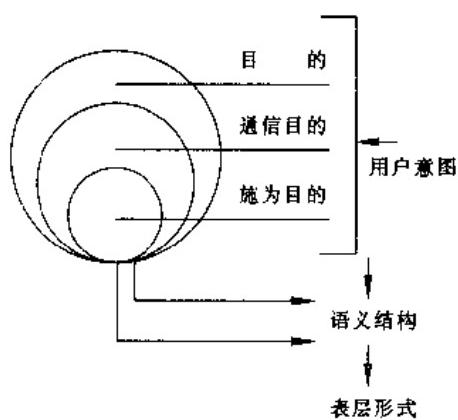


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

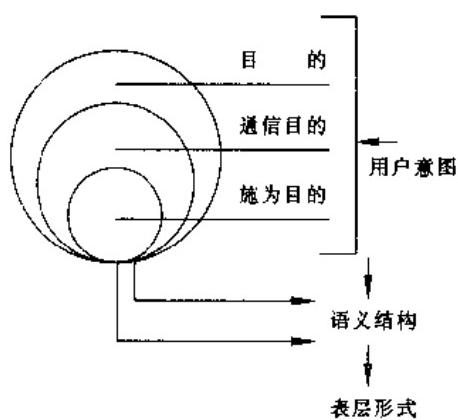


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削减冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

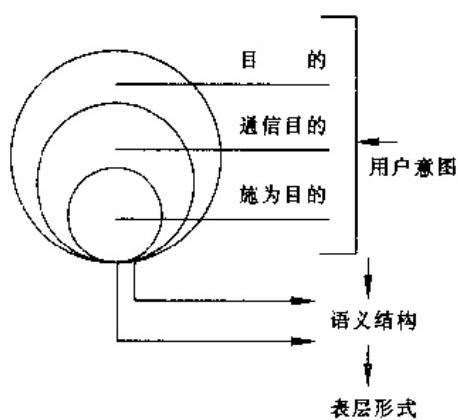


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

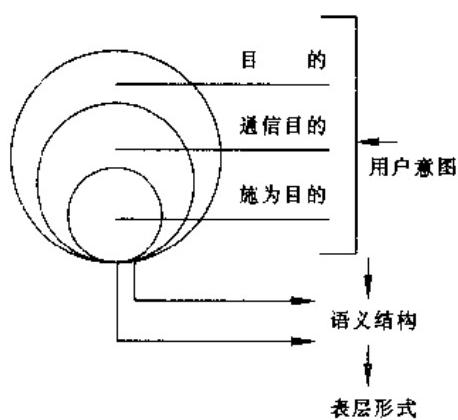


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

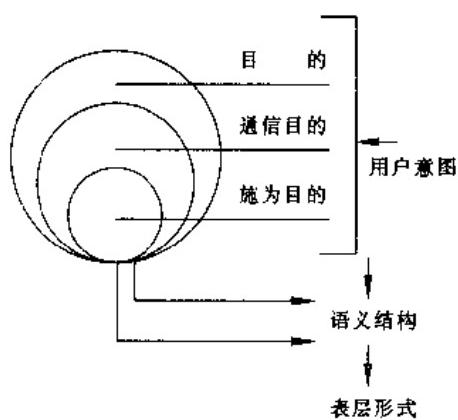


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外

言和口语的词汇等,都可以不在系统的考虑范围之内。这同样也意味着文本的规划具有十分重要的意义,因为写作时,作者通常会花大量的时间和精力推敲文章的内容和结构。另外,从实现环境而言,生成系统通常以文字的方式在计算机荧屏上输出它的结果,这也使得书面文本的生成要相对简单一些。

#### 4. 生成系统的任务和要求

从上面的分析中可知,最终生成的文本应该满足以下要求:

- (1)生成的文本不是事先封装于系统中的;
- (2)生成的文本所基于的语义表示应是可变化的,由程序动态生成的;
- (3)生成的文本应是连贯的,前后句间应存在语义和语法上的相关性;
- (4)生成的文本应具有正确的表层形式,是可理解的;

### 三、生成系统结构

自然语言生成系统一般包括两个部分:

- (1)依据用户意图确定输出的语义,称为“规划模块”;
- (2)把信息转化为正确的自然语言形式,称为“实现模块”,如图 12-11 所示。

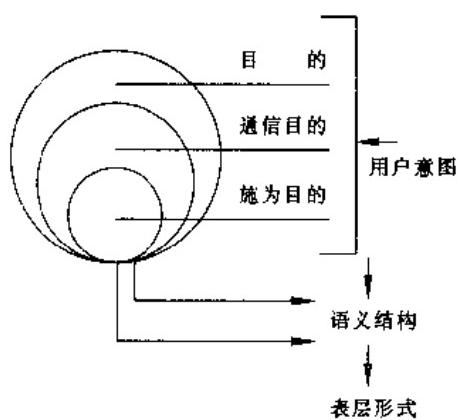


图 12-11 生成系统划分

两个模块间的通信方式与执行顺序决定了生成系统的结构。

#### 1. 串行结构

串行式系统结构是文本规划的传统做法。其处理模型包括两个阶段:第一阶段,称为“宏规划”,确定待输出的内容和结构,选择输出的概念,定义彼此间的关系,并提供后一阶段进行句法结构和词汇选择所需的判定信息。第二阶段,称为“微规划”,依据上下文语境、句子长度、风格等因素,削冗余信息,生成指称表述,调整待生成的语句,输出表层字符串。前者面向人的思维与知识,独立于目标语言,定义后者的输入;后者面向具体的应用语言,依赖于生成系统内部使用的语义表示机制。

在实际系统中,上述两个阶段还可继续细分为若干不同的处理层次。全部规划在实现模块调用之前完成,规划任务一旦结束,对于实现模块而言,规划器就没有任何意义了。它的结构框图如图 12-12 所示

这种串行体系结构体现的观点是,语言学知识与非语言学因素,在生成过程中可以截然分开,规划模块在生成语义表示时不参考所使用的特定语言,确定输出语句的结构和词汇时也无须回溯其概念内涵,认为语言学知识应该而且能够限制在一个模块中,负责表层语法规则所要求的形式与内容的实现,而把所有一般的推理留给其他的模块。

这种简单分割,在某种程度上是为了适应当前的计算机技术而设计的,不能真正模拟人类生成语言的过程,因为语言潜在地反映了社会的文化背景和一个民族特有的思维方式,宏规划器输出的语义表示必然应受到微规划器中语言知识的制约。

#### 2. 并行结构

生成过程需要两种形式的规划。某些规划任务应该在调用实现模块之前全部完成,而另外